The Virgin Computer Games Series

GAMES FOR YOUR BBC MICRO

£££££'s of entertaining games for only £2.95

GAMES FOR YOUR BBC MICRO

By Alex Gollner

First published in Great Britain in 1983 by Virgin Books Ltd, 61-63 Portobello Road,London W11 3DD.

Copyright © 1983 Interface/Virgin Books

ISBN 0 907080 83 9

All rights reserved. No part of this book may be reproduced in any form or by any means without prior permission from the publisher.

Printed and bound in Great Britain by Richard Clay (The Chaucer Press) Ltd, Suffolk.

Production services by Book Production Consultants, Cambridge.

Designed by Ray Hyden.

Illustrated by Sue Walliker.

Typeset by Portobello Typesetting.

Distributed by Hamlyn Paperbacks.

TIM HARTNELL — SERIES EDITOR

Tim Hartnell is a leading computer expert and journalist, who has contributed extensively to the Technical Consumer Press. He is also the author of several books including *Getting Acquainted With Your ZX81*, *Let Your BBC Micro Teach You to Program* and *Programming Your ZX Spectrum*.

ALEX GOLLNER — THE AUTHOR

Alex Gollner is a 16-year-old, presently at school studying for his 'O' levels.

SUE WALLIKER — THE ILLUSTRATOR

Sue Walliker is a freelance illustrator.

ACKNOWLEDGEMENTS

The author would like to thank Tim Hartnell, Jeremy Ruston and Richard and Iris Gollner.

Editor's Introduction

Your computer is waiting to challenge you. Moving graphics games, brain stretchers, word games and puzzles are all here and ready to entertain you.

A wide variety of games are included in this book. The programs have been written by some of the most talented young programmers working in this country at the moment, and represent a variety of approaches to solving programming problems.

An examination of the listings should teach you many tricks and techniques to apply to your own programming. And once you have mastered the programs in their present form, you might want to try your hand at improving them. There is no such thing as a 'perfect program', so these games are sure to benefit from your programming skill.

All that now remains is for you to turn the page and enter the programs. I can only hope that you enjoy playing the games as much as we did when preparing this volume.

Tim Hartnell, series editor
London
March 1983

Author's Introduction

Welcome to *Games for Your BBC Micro*. The programs included in this volume range from futuristic arcade games, such as 3D Terror Chase and Maze-Man, to traditional games such as Pontoon and Draughts. There is also a set of graphics to demonstrate the best features of your BBC Micro.

There are many examples of 'structured programming' in this book; they include modules, called procedures, which carry out separate tasks within each program. To help you learn from the programs, there are many REM statements and spare lines. These break the programs up so that you can see what is happening; they also employ useful programming hints and tips which can be applied to many other computer programs.

I hope you will input these programs and, having played and enjoyed them, learn how to improve them, so that they become your own individual programs. They all work in the BBC Micro operating system EPROM 0.10 upwards. And now for the listings.

Alex Gollner
London
January 1983

TO STEPHEN LOUGHRAN AND ANNA BOYLE

CONTENTS

Editor's Introduction 4

Author's Introduction 5

3D Terror Chase 8
React 21
Maze-Man 25
Pontoon 36
Tight Squeeze 46
Draughts 49
Life-Change Units 58
Spider and Fly 68
Magic Square 72
Dr Watson 75
Dr Audio 77
Nim 79
Chain 81
GCOL 83
Sound 84
Knightsbridge 87
GCOL—Tunnel 93
Moire 95
Square 97
Invader Blaster 98
Pac-Trac 105

How to Write Better Programs 110
Glossary 117

3D TERROR CHASE

Deep within an ancient castle on the foggy side of Mount Nausea dwell three paranoid ghosts. You have blundered into their haunt. Sweating with fear, you desperately search for freedom as they relentlessly close in on you.

When you run the program you'll see on the right of the screen a view of the corridor ahead — there will be doors in one, two or three of the walls facing you. In the top left-hand corner of the screen you will see a square. This is a map of the castle which shows where the ghosts (white dots) are in relation to yourself (a blue dot).

A rising column of red, starting at the bottom left-hand corner, represents time. You have to escape before the column hits the map from the near side of the castle to the far side on the map. When you reach the far side the game is over — you have won, successfully eluding the ghosts.

The control keys you should use are "Z" to go through the left-hand door, if it exists in the current corridor; "X" to go through the right-hand door; "L" to move ahead; "J" for teleporting, which carries you to another part of the castle instantaneously. This can only be used once in a game and is for a moment of dire desperation only. To slow down the game, increase the value of DEL at the start of line 60.

```
  10 ON ERROR RUN
  20 REM 3D Terror Chase...
  30
  40
  50 ENVELOPE 3,1,5,2,4,1,0,-1,1,-1,0,0
,120,10
  60 *FX 15
  70 DEL=5
  80 MODE 2
  90 VDU 23;8202;0;0;0;
 100 PROCINIT
 110 REPEAT
 120 PROCtime
 130 PROCmove
 140 PROCghosts
 150 UNTIL TIME>6000 OR (P%DIV28)=25 OR
 W%
 160 PROCend
 170 *FX 15,1
 180 END

 190 DEF PROCINIT
 200 LOCAL T%,G%,A$,B%,J%,X%,Y%,G,H%,Z%
 210 *FX 9
 220 REM DATA FOR MAZE FOLLOWS
 230 DATA ****************************
 240 DATA *.00.......000...00....000.*
 250 DATA *.00..000..000...0000...00.*
 260 DATA *.....000.......000....000.*
 270 DATA *..00..00.00.00.000000...0.*
 280 DATA *.00...00000............000.*
 290 DATA *....0.......0000..00......*
```

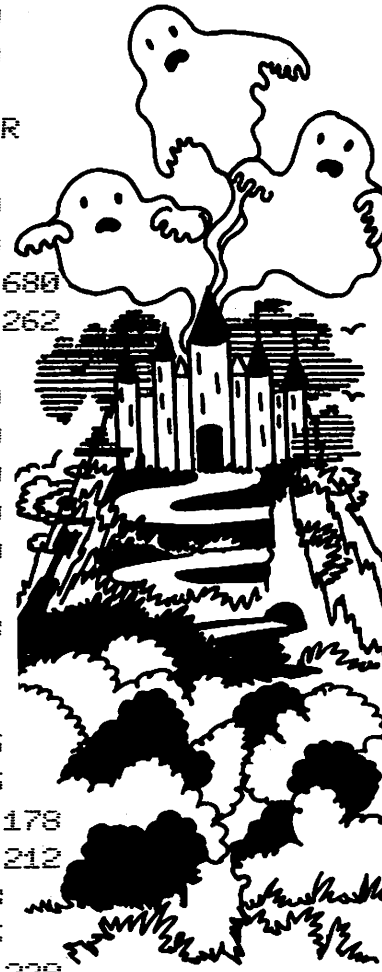
```
300 DATA *00..00...00....0...0...0..*
310 DATA *0..00...00....000..00....0*
320 DATA *....0....0..000.....0..0..*
330 DATA *00..00..000..000....00..0.*
340 DATA *............00...0....00.*
350 DATA *.000....000.....0....0....*
360 DATA *.000....000...00000...00..*
370 DATA *...0000.........0......0..*
380 DATA *..000.....0000..0......0..*
390 DATA *0..00..00..00...0000...00.*
400 DATA *..0000..0.....000.........*
410 DATA *....................0000*
420 DATA *000...000...000...000.....*
430 DATA *..........................*
440 DATA *...000...000...000...000..*
450 DATA *0..000.000000..0000000.000..*
460 DATA *...000...000...000...000.0*
470 DATA *...000...000...000...000..*
480 DATA *..........................*
490 DATA ****************************
500 OD%=1
510 DIM A%756
520 FOR T%=0 TO 26
530 READ A$
540 FOR G%=1 TO 28
550 ?(A%+T%*28+G%-1)=ASC(MID$(A$,G%,1)

560 NEXT G%
570 NEXT T%

580 REM WALLS CEILING AND FLOOR
590 GCOL 0,4
```

```
600 MOVE 300,0
610 MOVE 1200,0
620 PLOT 85,300,900
630 PLOT 85,1200,900
640 GCOL 0,6
650 MOVE 300,0
660 MOVE 1200,0
670 PLOT 85,600,350
680 PLOT 85,900,350
690 GCOL 0,7
700 MOVE 300,900
710 MOVE 1200,900
720 PLOT 85,600,750
730 PLOT 85,900,750

740 REM PLANKS
750 GCOL 0,0
760 MOVE 600,350
770 DRAW 900,350
780 DRAW 900,750
790 DRAW 600,750
800 DRAW 600,350
810 MOVE 300,900
820 DRAW 600,750
830 MOVE 1200,900
840 DRAW 900,750
850 B%=RND(9)*100-100
860 FOR T%=0 TO 800 STEP 100
870 MOVE 300+T%,0
880 DRAW 600+T% DIV 3,350
890 FOR J%=1 TO 2
900 G=RND(1)
```

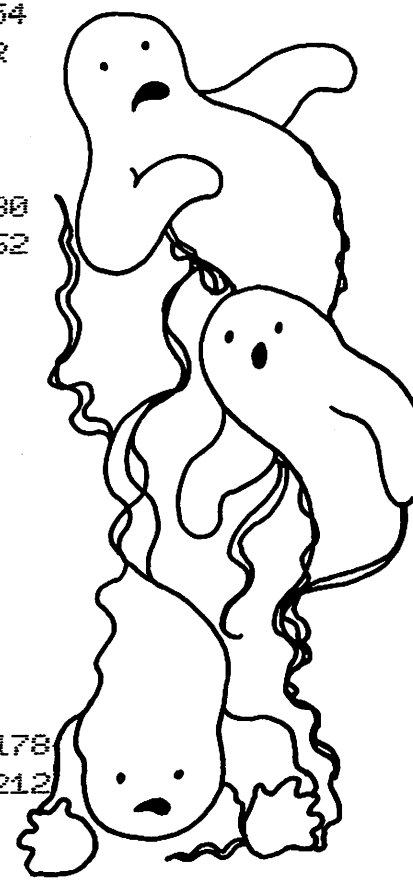
```
  910 Y%=350*G
  920 X%=(300+T% DIV 3-T%)*G+300+T%
  930 MOVE X%,Y%
  940 H%=T%+100
  950 Z%=(300+H% DIV 3-H%)*G+300+H%
  960 DRAW Z%,Y%
  970 IF T%=B% AND J%=2 O%=X%:P%=Y%:Q%=Z
%
  980 IF T%=B% AND J%=1 L%=X%:M%=Y%:N%=Z
%
  990 NEXT J%
 1000 NEXT T%

 1010 REM FAR DOOR
 1020 MOVE L%,M%
 1030 MOVE N%,M%
 1040 PLOT 85,O%,P%
 1050 PLOT 85,Q%,P%
 1060 GCOL 0,0
 1070 MOVE 1200,0
 1080 DRAW 900,350
 1090 GCOL 0,15
 1100 MOVE 680,354
 1110 MOVE 820,354
 1120 PLOT 85,680,660
 1130 PLOT 85,820,660
 1140 GCOL 0,12
 1150 MOVE 680,350
 1160 DRAW 680,660
 1170 DRAW 820,660
 1180 DRAW 820,350
 1190 MOVE 800,508
```

```
1200 PLOT 1,-8,0
1210 PLOT 1,0,-8
1220 PLOT 1,8,0
1230 PLOT 1,0,8
1240 MOVE 700,485
1250 MOVE 740,485
1260 PLOT 85,700,370
1270 PLOT 85,740,370
1280 MOVE 760,485
1290 MOVE 800,485
1300 PLOT 85,760,370
1310 PLOT 85,800,370
1320 MOVE 700,530
1330 MOVE 740,530
1340 PLOT 85,700,640
1350 PLOT 85,740,640
1360 MOVE 760,530
1370 MOVE 800,530
1380 PLOT 85,760,640
1390 PLOT 85,800,640
1400 GCOL 0,3

1410 REM LIGHT RAYS
1420 FOR T%=0 TO 90 STEP 5
1430 MOVE 750,820
1440 H%=100+RND(100)
1450 PLOT 1,SIN(RAD(T%+135))*H%,COS(RAD
(T%+135))*H%
1460 NEXT T%
1470 REM LIGHT SHADE
1480 GCOL 0,0
1490 MOVE 750,850
```

```
1500 PLOT 1,0,-30
1510 PLOT 1,-40,-40
1520 PLOT 81,80,0
1530 MOVE 738,850
1540 PLOT 1,22,0
1550 REM LEFT DOOR
1560 GCOL 0,13
1570 MOVE 400,700
1580 MOVE 400,128
1590 PLOT 85,516,680
1600 PLOT 85,516,262
1610 GCOL 0,10
1620 MOVE 400,120
1630 DRAW 400,700
1640 DRAW 516,680
1650 DRAW 516,250
1660 MOVE 500,510
1670 PLOT 1,0,8
1680 PLOT 1,-8,-8
1690 PLOT 1,0,-8
1700 PLOT 1,8,4
1710 MOVE 420,465
1720 MOVE 450,475
1730 PLOT 85,420,178
1740 PLOT 85,450,212
1750 MOVE 466,478
1760 MOVE 500,486
1770 PLOT 85,466,228
1780 PLOT 85,500,266
1790 MOVE 420,515
1800 MOVE 450,520
1810 PLOT 85,420,664
```



```
1820 PLOT 85,450,660
1830 MOVE 466,524
1840 MOVE 500,534
1850 PLOT 85,466,660
1860 PLOT 85,500,654
1870 REM RIGHT DOOR
1880 GCOL 0,14
1890 MOVE 1100,700
1900 MOVE 1100,128
1910 PLOT 85,984,680
1920 PLOT 85,984,262
1930 GCOL 0,11
1940 MOVE 1100,120
1950 DRAW 1100,700
1960 DRAW 984,680
1970 DRAW 984,250
1980 MOVE 1084,494
1990 PLOT 1,0,-8
2000 PLOT 1,-8,8
2010 PLOT 1,0,8
2020 PLOT 1,8,-4
2030 MOVE 1080,465
2040 MOVE 1050,475
2050 PLOT 85,1080,178
2060 PLOT 85,1050,212
2070 MOVE 1034,478
2080 MOVE 1000,486
2090 PLOT 85,1034,228
2100 PLOT 85,1000,266
2110 MOVE 1080,515
2120 MOVE 1050,520
2130 PLOT 85,1080,664
```



```
 2140 PLOT 85,1050,660
 2150 MOVE 1034,524
 2160 MOVE 1000,534
 2170 PLOT 85,1034,660
 2180 PLOT 85,1000,654
 2200 DIM U%(2)
 2210 REPEAT
 2220 B%=RND(756)-1
 2230 UNTIL A%?B%=46
 2240 U%(0)=B%
 2250 REPEAT
 2260 B%=RND(756)-1
 2270 UNTIL A%?B%=46 AND B%<>U%(0)
 2280 U%(1)=B%
 2290 REPEAT
 2300 B%=RND(756)-1
 2310 UNTIL A%?B%=46 AND B%<>U%(1) AND B
%<>U%(0)
 2320 U%(2)=B%
 2330 REPEAT
 2340 B%=RND(56)-1
 2350 UNTIL A%?B%=46 AND B%<>U%(0) AND B
%<>U%(1) AND B%<>U%(2)
 2360 P%=B%
 2370 TIME=0
 2380 GCOL 0,2
 2390 MOVE 0,690
 2400 DRAW 224,690
 2410 DRAW 224,914
 2420 DRAW 0,914
 2430 DRAW 0,690
 2440 MOVE 0,694
```

```
 2450 PLOT 1,224,0
 2460 MOVE 0,910
 2470 PLOT 1,224,0
 2480 C%=TRUE
 2490 V%=0
 2500 ENDPROC
 2510 DEFPROCtime
 2520 LOCAL K%
 2530 K%=TIME DIV 35
 2540 IF K%=V% ENDPROC
 2550 GCOL 0,1
 2560 IF K% MOD 2=1 MOVE-8,K%*4 ELSE MOV
E 0,K%*4
 2570 PLOT 21,250,K%*4
 2580 V%=K%
 2590 ENDPROC
 2600 DEF PROCmove
 2610 LOCAL X%,Y%,B%,K%
 2620 GCOL 0,0
 2630 X%=(P% MOD 28)*8
 2640 Y%=(P% DIV 28)*8
 2650 PLOT 69,X%,Y%+690
 2660 PLOT 69,X%,Y%+694
 2670 X%=TIME
 2680 B%=INKEY(DEL) AND &DF
 2690 IF B%=90 AND ?(A%+P%-1)=46 P%=P%-1
 2700 REPEAT
 2710 PROCtime
 2720 UNTIL X%+DEL<TIME
 2730 IF B%<>223 SOUND 17,-15,200,4
 2740 *FX 15,1
 2750 IF B%=88 AND ?(A%+P%+1)=46 P%=P%+1
```

```
 2760 IF B%=76 AND ?(A%+P%+28)=46 P%=P%+
28
 2770 IF B%=74 AND C% REPEAT B%=RND(378)
-1:UNTIL A%?B%=46:P%=B%:C%=FALSE
 2780 GCOL 0,4
 2790 X%=(P% MOD 28)*8
 2800 Y%=(P% DIV 28)*8
 2810 W%=FALSE
 2820 IF POINT(X%,Y%+690)<>0 W%=TRUE
 2830 PLOT 69,X%,Y%+690
 2840 PLOT 69,X%,Y%+694
 2850 IF P%=OD% ENDPROC
 2860 OD%=P%
 2870 IF ?(A%+P%-1)=46 VDU 19,10;0;0;19,
13,RND(7);0; ELSE VDU 19,10,4;0;0;19,13,
4;0;
 2880 IF ?(A%+P%+1)=46 VDU 19,11;0;0;19,
14,RND(7);0; ELSE VDU 19,11,4;0;0;19,14,
4;0;
 2890 IF ?(A%+P%+28)=46 VDU 19,12;0;0;19
,15,RND(7);0; ELSE VDU 19,12,4;0;0;19,15
,4;0;
 2900 ENDPROC

 2910 DEF PROCghosts
 2920 LOCAL T%,X%,Y%,L%,M%,D%
 2930 X%=P% MOD 28
 2940 Y%=P% DIV 28
 2950 FOR T%=0 TO 2
 2960 L%=U%(T%) MOD 28
 2970 M%=U%(T%) DIV 28
 2980 D%=U%(T%)
```

```
2990 GCOL 0,0
3000 PLOT 69,L%*8,M%*8+690
3010 PLOT 69,L%*8,M%*8+694
3020 F%=0
3030 IF ?(A%+D%+28)=46 F%=1
3040 IF ?(A%+D%-28)=46 F%=F% OR 4
3050 IF ?(A%+D%+1)=46 F%=F% OR 8
3060 IF ?(A%+D%-1)=46 F%=F% OR 2
3070 R%=SGN(X%-L%)
3080 S%=SGN(Y%-M%)
3090 G%=0
3100 IF R%=1 G%=8
3110 IF R%=-1 G%=G% OR 2
3120 IF S%=1 G%=G% OR 1
3130 IF S%=-1 G%=G% OR 4
3140 K%=F% AND G%
3150 IF K%=0 K%=F%
3160 REPEAT
3170 R%=RND(4)-1
3180 UNTIL (2^R% AND K%)<>0
3190 IF R%=0 M%=M%+1
3200 IF R%=1 L%=L%-1
3210 IF R%=2 M%=M%-1
3220 IF R%=3 L%=L%+1
3230 U%(T%)=L%+M%*28
3240 IF U%(T%)=P% W%=TRUE
3250 GCOL 0,7
3260 PLOT 69,L%*8,M%*8+690
3270 PLOT 69,L%*8,M%*8+694
3280 NEXT T%
3290 ENDPROC
3300 DEF PROCend
```

```
 3310 ENVELOPE 1,2,5,2,10,5,2,5,10,1,126
,-2,90,126
 3320 SOUND 17,1,200,255
 3330 VDU 17,2,30
 3340 PRINT "The time is ";TIME DIV 100;
" and "
 3350 IF TIME>6000 PRINT"so you ran out
of it...":ENDPROC
 3360 IF(P% DIV 28)=25 PRINT "you did it
! Hearty  congrats.":ENDPROC
 3370 IF W% PRINT "the ghosts got you!"
 3380 *FX 9,3
 3390 SOUND 529,3,10,255
 3400 SOUND 530,3,15,255
 3410 SOUND 531,3,20,255
 3420 *FX 10,3
 3430 GCOL 0,8
 3440 DIM P%(12,1)
 3450 FOR T%=1 TO 12
 3460 P%(T%,0)=T%*100
 3470 P%(T%,1)=-RND(30)
 3480 NEXT
 3490 REPEAT
 3500 FOR T%=1 TO 12
 3510 PLOT 69,P%(T%,0),P%(T%,1)
 3520 P%(T%,1)=P%(T%,1)+4
 3530 A%=RND(3)
 3540 IF A%=1 P%(T%,0)=P%(T%,0)+8
 3550 IF A%=2 P%(T%,0)=P%(T%,0)-8
 3560 NEXT
 3570 UNTIL P%(1,1)>1023
 3580 ENDPROC
```

REACT

This game of sound, lights and vision tests your reactions. The idea is to see how good you are at recognising sounds, at finding letters on the keyboard and at finding letters that show up on the screen. The program starts with a menu, so you can decide which of the three reaction tests you want to try.

```
   10 REM REACT
   20 ON ERROR RUN
   30 ENVELOPE 1,1,110,120,130,1,10,100,
0,0,0,0,100,0
   40 MODE 7
   44 REM TURN OFF CURSOR
   45 VDU 23;8202;0;0;0;
   50 FOR S=5 TO 6
   60 PRINT TAB(13,S-3) CHR$(129);CHR$(1
57);CHR$(141);CHR$(132);"REACT    ";CHR$(
156)
   70 NEXT
   80 PRINT'TAB(6)"1. . . . . FOR S
OUND TEST"
   90 PRINT'TAB(6)"2. . . . . FOR CHARA
CTER TEST"
  100 PRINT'TAB(6)"3. . . . . FOR POSI
TION TEST"
  110 PRINT'TAB(6)"4. . . . . TO LEAV
E PROGRAM"
  120 REM ANSWER
  130 REPEAT
  140 A$=GET$
  150 UNTIL A$="1" OR A$="2" OR A$="3" O
R A$="4"
```

```
  160 IF A$="4" END
  170 IF A$="3" PROCpos
  180 IF A$="2" PROCchar
  190 IF A$="1" PROCsound
  200 RUN
  210 DEF PROCpos
  220 CLS
  250 REM ENVELOPE-INFINITE LENGTH
  260 SOUND 1,1,0,255
  270 REM PRINT 200 RANDOM BLOCKS
  280 FOR D=1 TO 200
  290 PROCrnd
  300 VDU 31,X,Y,W,255
  310 NEXT
  320 PROCwait
  330 PROCrnd
  340 VDU 31,X,Y,W,Z
  350 TIME=0
  360 REM KEEP CHECKING FOR INPUT-UNTIL
INPUT = CHARACTER ON SCREEN
  370 REPEAT
  380 WZ=GET
  390 UNTIL WZ=Z
  400 PRINT TAB(0,1) "You took ";TIME/10
0;" Seconds"
  410 REM TURN OFF ENVELOPE
  420 SOUND 17,0,0,0
  430 S=GET
  440 PROCpos
  450 DEF PROCrnd
  460 W=128+RND(7)
  470 X=RND(39)
  480 Y=RND(24)
```

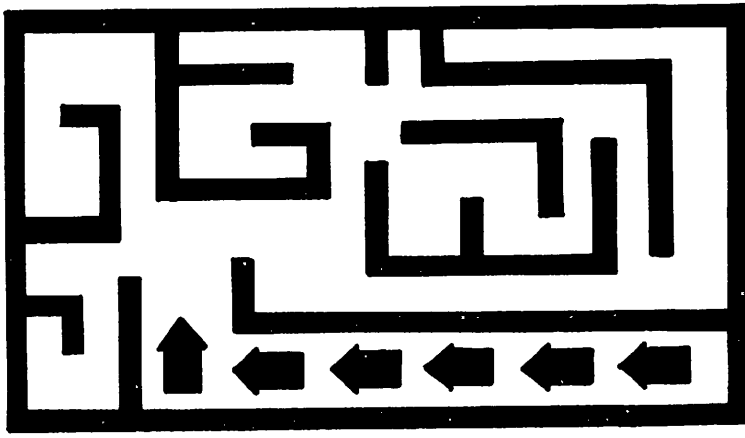
```
  490 Z=RND(26)+65
  500 ENDPROC
  510 DEF PROCwait
  520 FOR W=1 TO 2000+RND(8000)
  530 NEXT
  540 ENDPROC
  550 DEF PROCsound
  560 CLS
  570 PROCkey
  580 PROCwait
  590 ZX=RND(20)
  600 PROCso(ZX)
  610 VDU 31,20,13,255
  620 TIME=0
  630 REPEAT
  640 REPEAT
  650 A=GET
  660 UNTIL A>37 OR A<58
  670 A=A-37
  680 PROCso(A)
  690 UNTIL A=ZX
  700 PRINT TAB(0,1) "You took ";TIME/10
0;" Seconds"
  710 C=GET
  720 PROCsound
  730 ENDPROC
  740 DEF PROCkey
  750 *KEY 0 &
  760 *KEY 1 '
  770 *KEY 2 (
  780 *KEY 3 )
  790 *KEY 4 *
  800 *KEY 5 +
```

```
 810 *KEY 6 ,
 820 *KEY 7 -
 830 *KEY 8 .
 840 *KEY 9 /
 850 ENDPROC
 860 DEF PROCso(S)
 870 REM CHORD MAKER
 880 SOUND 17,-10,50+(S*10),10
 890 SOUND 18,-10,52+(S*10),10
 900 SOUND 19,-10,54+(S*10),10
 910 ENDPROC
 920 DEF PROCchar
 930 CLS
 940 SOUND 17,1,-20,255
 950 Z=RND(96)+32
 960 PROCwait
 961 VDU 28,10,18,30,12
 970 ?&36D=Z
 972 CLS
 980 TIME=0
 990 REPEAT
1000 W=GET
1010 SOUND 2,-15,1,1
1020 UNTIL W=Z
1030 SOUND 17,-10,100,1
1035 VDU 26
1040 PRINT TAB(0,1) "You took ";TIME/10
0;" Seconds"
1045 ?&36D=32
1050 S=GET
1060 PROCchar
1070 ENDPROC
```

MAZE-MAN

You are trapped in a bright-green maze and hunted by ferocious multi-coloured snipe. The floor of the maze is littered with red anti-matter capsules. Run over one, and your score is reduced. The aim is to get a score of 0, though you'll be lucky to get one lower than 45. The cursor keys are your controls and you must move in the direction of the arrows on those keys. You need only tap the keys — holding them down will only confuse the poor computer! If no key is pressed, you will continue to go in the current direction.

In the corners of the maze you will find yellow anti-anti-matter (sic) capsules. Here even the snipe change: instead of being multi-coloured, they emit black and white flashes. They can then be eaten by *you*. If you manage to consume one of the snipe under these conditions, you will be awarded an extra life. During the game, you are allowed to use three lives. The speed of the game may be decreased by increasing the values of the descent (D% in line 90).



```
 30 REM Maze-man
 40 *TV 255
 50 LOW%=150
 60 D%=0
 70 REPEAT
 80 MODE 2
 90 PROCscreen
100
110 REPEAT
120 PROCmove_man
130 PROCmove_snipe
140 COLOUR 4
150 PRINT TAB(6,1);N%;"    "
160 UNTIL N%=0 OR L%=0
170 TIME=0
180 REPEAT UNTIL TIME>100
190 MODE 7
200 IF L%=0 THEN PROClose
210 IF N%=0 THEN PROCwin
220 IF N%<LOW% THEN LOW%=N%
230 *FX 15,1
240 UNTIL GET<>13
250 *FX 12
260 *FX 4
270 END
280
290 DEF PROCscreen
300 LOCAL A%,T%,G%,A$,B$
310 VDU 23;8202;0;0;0;
320 CLS
330 RESTORE
340 COLOUR 4
```

```
350 @%=0
360 DATA FF,FF,FF,FF,FF,FF,FF,FF
370 DATA 00,00,08,1C,08,00,00,00
380 DATA 3C,7E,FF,FF,FF,FF,7E,3C
390 DATA 3C,7C,F8,F0,F0,F8,7C,3C
400 DATA 3C,3E,1F,0F,0F,1F,3E,3C
410 DATA 00,00,C3,E7,FF,FF,7E,3C
420 DATA 3C,7E,FF,FF,E7,C3,00,00
430 DATA 18,3C,7E,5A,7E,66,7E,5A
440 DATA 00,00,00,18,18,00,00,00
450 PRINT "Low score:"LOW%'"Score:"'
460 FOR T%=224 TO 232
470 VDU 23,T%
480 FOR G%=0 TO 7
490 READ A$
500 VDU EVAL("&"+A$)
510 NEXT
520 NEXT
530 DATA "00000000000000000000"
540 DATA "0*.....0.....*0"
550 DATA "0.0.000.0.000.0.0"
560 DATA "0.0.000.0.000.0.0"
570 DATA "0..............0"
580 DATA "0.0.0.00000.0.0.0"
590 DATA "0...0...0...0...0"
600 DATA "000.000.0.000.000"
610 DATA "   0.0.......0.0   "
620 DATA "000.0.00000.0.000"
630 DATA ".....0     0....."
640 DATA "000.0.00000.0.000"
650 DATA "   0.0.......0.0   "
660 DATA "000.0.00000.0.000"
```

```
670 DATA "0.....0.....0"
680 DATA "0.0.000.0.000.0.0"
690 DATA "0.............0"
700 DATA "000.0.00000.0.000"
710 DATA "0...0...0...0...0"
720 DATA "0.00000.0.00000.0"
730 DATA "0*...........*0"
740 DATA "00000000000000000"
750 FOR T%=1 TO 22
760 READ A$
770 FOR G%=1 TO 17
780 B$=MID$(A$,G%,1)
790 IF B$=" " THEN VDU 32
800 IF B$="." THEN VDU 17,1,225
810 IF B$="0" THEN VDU 17,2,224
820 IF B$="*" THEN VDU 17,3,232
830 NEXT
840 PRINT
850 NEXT
860 *FX 12,1
870 *FX 11,1
880 *FX  4,1
890 *FX  9,5
900 *FX 10,5
910 *KEY 0"*FX 12|M*FX 4|M"
920 COLOUR 3
930 X%=7:Y%=11
940 L%=3
950 Z%=227
960 N%=150
970 Q%=0
980 R%=FALSE
```

```
  990 VDU 31,X%,Y%,31,18,7,231,31,18,9,2
31,31,18,11,231
 1000 DIM U%2,V%2,K%2
 1010 FOR T%=0 TO 2
 1020 U%?T%=7+T%
 1030 V%?T%=15
 1040 K%?T%=225
 1050 NEXT
 1060 ENDPROC
 1070
 1080 DEF PROCmove_man
 1090 LOCAL A%,T%
 1100 VDU 17,7,31,X%,Y%,226
 1110 TIME=0
 1120 A%=INKEY(D%)
 1130 IF A%<136 OR A%>139 THEN A%=Q%
 1140 REPEAT UNTIL TIME>D%
 1150 *FX 15,1
 1160 *FX 15,6
 1170 IF A%=139 THEN PROCup
 1180 IF A%=138 THEN PROCdown
 1190 IF A%=137 THEN PROCright
 1200 IF A%=136 THEN PROCleft
 1210 Q%=A%
 1220 VDU 31,X%,Y%,Z%
 1230 ENDPROC
 1240
 1250 DEF PROCup
 1260 C%=FNread(X%,Y%-1)
 1270 IF C%=224 THEN A%=Q% : ENDPROC
 1280 VDU 31,X%,Y%,32
 1290 Y%=Y%-1
```

```
 1300 IF C%=225 OR C%=232 THEN N%=N%-1:S
OUND 1,-15,10,2 ELSE SOUND 1,-15,200,2
 1310 IF C%=232 THEN PROCchange
 1320 Z%=229
 1330 ENDPROC
 1340
 1350 DEF PROCdown
 1360 C%=FNread(X%,Y%+1)
 1370 IF C%=224 THEN A%=0% : ENDPROC
 1380 VDU 31,X%,Y%,32
 1390 Y%=Y%+1
 1400 IF C%=225 OR C%=232 THEN N%=N%-1:S
OUND 1,-15,10,2 ELSE SOUND 1,-15,200,2
 1410 IF C%=232 THEN PROCchange
 1420 Z%=230
 1430 ENDPROC
 1440
 1450 DEF PROCright
 1460 C%=FNread(X%+1,Y%)
 1470 IF C%=224 THEN A%=0% : ENDPROC
 1480 VDU 31,X%,Y%,32
 1490 X%=X%+1
 1500 IF C%=225 OR C%=232 THEN N%=N%-1:S
OUND 1,-15,10,2 ELSE SOUND 1,-15,200,2
 1510 IF C%=232 THEN PROCchange
 1520 Z%=227
 1530 IF X%>16 THEN X%=0
 1540 ENDPROC
 1550
 1560 DEF PROCleft
 1570 C%=FNread(X%-1,Y%)
 1580 IF C%=224 THEN A%=0% : ENDPROC
```

Taped (To check)

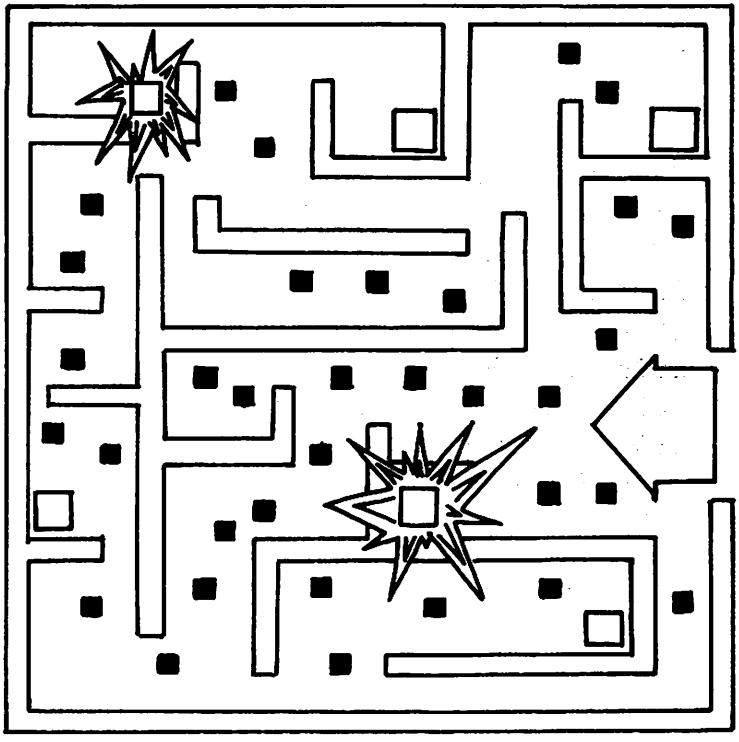

```
 1590 VDU 31,X%,Y%,32
 1600 X%=X%-1
 1610 IF C%=225 OR C%=232 THEN N%=N%-1:S
OUND 1,-15,10,2 ELSE SOUND 1,-15,200,2
 1620 IF C%=232 THEN PROCchange
 1630 Z%=228
 1640 IF X%<0 THEN X%=16
 1650 ENDPROC
 1660
 1670 DEF FNread(X%,Y%)
 1680 LOCAL A%,C%
 1690 A%=135
 1700 VDU 31,(X%+17) MOD 17,Y%
 1710 C%=((USR(&FFF4) AND &FFFF) DIV &10
0)
 1720 IF C%<128 THEN =C%
 1730=(C% MOD 32)+224
 1740
 1750 DEF FNdir(X%,Y%)
 1760 LOCAL D%
 1770 D%=0
 1780 IF FNread(X%-1,Y%)<>224 THEN D%=D%
+1
 1790 IF FNread(X%,Y%-1)<>224 THEN D%=D%
+2
 1800 IF FNread(X%,Y%+1)<>224 THEN D%=D%
+4
 1810 IF FNread(X%+1,Y%)<>224 THEN D%=D%
+8
 1820=D%
 1830
 1840 DEF PROCmove_snipe
```

```
 1850 LOCAL T%,J%,I%,H%
 1860 IF R% THEN S%=S%-1 : IF S%=0 THEN
R%=FALSE
 1870 FOR T%=0 TO 2
 1880 I%=U%?T%
 1890 J%=V%?T%
 1900 IF K%?T%=225 THEN COLOUR 1 ELSE CO
LOUR 3
 1910 VDU 31,I%,J%,K%?T%
 1920 PROCnew_direction
 1930 IF I%>16 THEN I%=0
 1940 IF I%<0  THEN I%=16
 1950 U%?T%=I%
 1960 V%?T%=J%
 1970 IF R% THEN COLOUR 8 ELSE COLOUR 3
 1980 K%?T%=FNread(U%?T%,V%?T%)
 1990 O%=FALSE
 2000 IF (K%?T%<>225 AND K%?T%<231 AND K
%?T%<>32) OR (I%=X% AND J%=Y%) THEN PROC
what_next
 2010 IF K%?T%=231 THEN K%?T%=225
 2020 VDU 31,U%?T%,V%?T%,231
 2030 IF O% THEN T%=5
 2040 NEXT
 2050 ENDPROC
 2060
 2070 DEF PROCnew_direction
 2080 LOCAL D%,L%,M%,H%,P%
 2090 D%=FNdir(I%,J%)
 2100 IF R%=FALSE THEN L%=SGN(X%-I%) : M
%=SGN(Y%-J%) ELSE L%=SGN(I%-X%) : M%=SGN
(J%-Y%)
```

```
 2110 H%=0
 2120 IF L%=-1 THEN H%=H%+1
 2130 IF L%=+1 THEN H%=H%+8
 2140 IF M%=-1 THEN H%=H%+2
 2150 IF M%=+1 THEN H%=H%+4
 2160 P%=H% AND D%
 2170 IF P%=1 THEN I%=I%-1 : ENDPROC
 2180 IF P%=2 THEN J%=J%-1 : ENDPROC
 2190 IF P%=4 THEN J%=J%+1 : ENDPROC
 2200 IF P%=8 THEN I%=I%+1 : ENDPROC
 2210 IF D%=1 THEN I%=I%-1 : ENDPROC
 2220 IF D%=2 THEN J%=J%-1 : ENDPROC
 2230 IF D%=4 THEN J%=J%+1 : ENDPROC
 2240 IF D%=8 THEN I%=I%+1 : ENDPROC
 2250 REPEAT
 2260 P%=D% AND (2^(RND(4)-1))
 2270 UNTIL P%
 2280 IF P%=1 THEN I%=I%-1 : ENDPROC
 2290 IF P%=2 THEN J%=J%-1 : ENDPROC
 2300 IF P%=4 THEN J%=J%+1 : ENDPROC
 2310 IF P%=8 THEN I%=I%+1 : ENDPROC
 2320 STOP
 2330
 2340 DEF PROClose
 2350 PRINT'CHR$(131);CHR$(141);TAB(10);
"G A M E  O V E R"
 2360 PRINT CHR$(130);CHR$(141);TAB(10);
"G A M E  O V E R"
 2370 PRINT'CHR$(132);"You have lost wit
h a score of ";N%
 2380 PRINT'''CHR$(129);"  Press 'return
' to restart..."
```

```
 2390 ENDPROC
 2400
 2410 DEF PROCwin
 2420 PRINT'CHR$(131);CHR$(141);TAB(10);
"G A M E  O V E R"
 2430 PRINT CHR$(130);CHR$(141);TAB(10);
"G A M E  O V E R"
 2440 PRINT'CHR$(132);"You won !!!"
 2450 PRINT''CHR$(129);"  Press 'return
' to restart..."
 2460 ENDPROC
 2470
 2480 DEF PROChit
 2490 VDU 31,18,L%*2+5,32
 2500 L%=L%-1
 2510 X%=0
 2520 Y%=13
 2530 K%?T%=32
 2540 SOUND 0,-15,2,20
 2550 O%=TRUE
 2560 ENDPROC
 2570
 2580 DEF PROCchange
 2590 SOUND 2,-15,255,10
 2600 R%=TRUE
 2610 S%=30
 2620 ENDPROC
 2630
 2640 DEF PROCgain
 2650 L%=L%+1
 2660 IF L%<10 THEN VDU 17,3,31,18,L%*2+
5,231
```

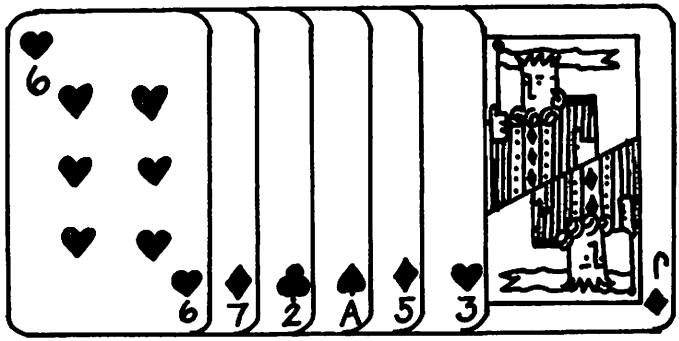
```
2670 K%?T%=32
2680 X%=0
2690 Y%=13
2700 SOUND 2,-15,100,70
2710 O%=TRUE
2720 ENDPROC
2730
2740 DEF PROCwhat_next
2750 IF R% THEN PROCgain ELSE PROChit
2760 ENDPROC
2770
```



PONTOON

The object of this game (based on the popular card game) is to get as close as possible to a score of 21. If you exceed 21, you have gone 'bust'. It is your turn first. The computer displays your first two cards, and you then have to decide whether to accept another card, in which case you 'twist', or to stay as you are, that is 'stick'. You can make the decision perhaps two or three more times. If your total exceeds 21, the computer will announce that you are 'bust', and you lose money.

Once you have decided to 'stick' with a score of less than 21, the computer deals its own hand. The computer ends up with a hand of cards, which is displayed, and will either go 'bust' (that is, get a score of over 21) or 'stick', in which case it will report which of you has won. You begin the game with $50 and the computer begins with $200. The idea is to win all the computer's money by beating it continuously.



```
  30 REM PONTOON

  40 MODE 2
  50 VDU 23;8202;0;0;0;
  60 VDU 19,0,4,0,0,0
  70 PROCchr

  80 MONEY=50

  90 CMON=200
 100 REM
 110 C1=RND(9)
 120 C2=RND(9)
 130 C3=RND(9)
 140 B=RND(4)+229
 150 COLOUR 2

 160 PROCcard
 170 PRINT TAB(0,1)"YOUR CARDS"
 180 PRINT TAB(0,29)"YOU:$";MONEY;TAB(1
0);"COMP:$";CMON
 190 VDU 31,2,2,235,239,239,239,236
 200 VDU 31,2,3,240,B,32,32,240
 210 VDU 31,2,4,240,32,32,32,240
 220 VDU 31,2,5,240,32,32,32,240
 230 VDU 31,2,6,240,32,C1+48,32,240
 240 VDU 31,2,7,240,32,32,32,240
 250 VDU 31,2,8,240,32,32,32,240
 260 VDU 31,2,9,240,32,32,B,240
 270 VDU 31,2,10,237,239,239,239,238

 280 SOUND 1,-15,200,1
```

```
290 PROCwait
300 B=RND(4)+229

310 VDU 31,8,2,235,239,239,239,236
320 VDU 31,8,3,240,B,32,32,240
330 VDU 31,8,4,240,32,32,32,240
340 VDU 31,8,5,240,32,32,32,240
350 VDU 31,8,6,240,32,C2+48,32,240
360 VDU 31,8,7,240,32,32,32,240
370 VDU 31,8,8,240,32,32,32,240
380 VDU 31,8,9,240,32,32,B,240
390 VDU 31,8,4,240,32,32,32,240
400 VDU 31,8,10,237,239,239,239,238

410 SOUND 1,-15,200,1
420 T0=C1+C2
430 COLOUR 3
440 PROCcheck
450 P=8
460 PRINT TAB(1,20) "  T-TWIST"
470 PRINT TAB(1,22) "  S-STICK"
480 REPEAT
490 REPEAT
500 A$=GET$
510 UNTIL A$="T" OR A$="S"
520 IF A$="S" THEN PROCstick
530 T0=T0+C3
540 B=RND(4)+229
550 P=P+6
560 COLOUR 2
570 IF P>15 THEN P=2:CLS:PROCcard
580 PRINT TAB(0,1)"YOUR CARDS"
```

```
590 PRINT TAB(0,29)"YOU:$";MONEY;TAB(1
0)"COMP:$";CMON
600 VDU 31,P,2,235,239,239,239,236
610 VDU 31,P,3,240,B,32,32,240
620 VDU 31,P,4,240,32,32,32,240
630 VDU 31,P,5,240,32,32,32,240
640 VDU 31,P,6,240,32,C3+48,32,240
650 VDU 31,P,7,240,32,32,32,240
660 VDU 31,P,8,240,32,32,32,240
670 VDU 31,P,9,240,32,32,B,240
680 VDU 31,P,10,237,239,239,239,238

690 SOUND 1,-15,200,1
700 C3=RND(9)
710 COLOUR 3
720 UNTIL T0>21
730 PRINT TAB(0,20) "    YOU BUST!"
740 PRINT TAB(0,22) "      I WIN"
750 PRINT TAB(0,24) "    THAT HAND"
760 SOUND 0,-15,10,4
770 MONEY=MONEY-25
780 CMON=CMON+25
790 PROCwait
800 PROCcheck
810 CLS
820 GOTO 100

830 DEF PROCstick
840 IF T0=21 THEN PROCpon
850 CC1=RND(9)
860 CLS
870 COLOUR 1
```

```
  880 PROCcard
  890 B=RND(4)+229
  900 PRINT TAB(0,29)"YOU:$";MONEY;TAB(1
0);"COMP:$";CMON
  910 PRINT TAB(0,1)"THE COMPUTER'S CARD
S"
  920 VDU 31,2,2,235,239,239,239,236
  930 VDU 31,2,3,240,B,32,32,240
  940 VDU 31,2,4,240,32,32,32,240
  950 VDU 31,2,5,240,32,32,32,240
  960 VDU 31,2,6,240,32,CC1+48,32,240
  970 VDU 31,2,7,240,32,32,32,240
  980 VDU 31,2,8,240,32,32,32,240
  990 VDU 31,2,9,240,32,32,B,240
 1000 VDU 31,2,10,237,239,239,239,238

 1010 SOUND 1,-15,100,1
 1020 CC2=RND(9)
 1030 CT=CC1+CC2
 1040 P=8
 1050 B=RND(4)+229
 1060 PROCwait

 1070 VDU 31,P,2,235,239,239,239,236
 1080 VDU 31,P,3,240,B,32,32,240
 1090 VDU 31,P,4,240,32,32,32,240
 1100 VDU 31,P,5,240,32,32,32,240
 1110 VDU 31,P,6,240,32,CC2+48,32,240
 1120 VDU 31,P,7,240,32,32,32,240
 1130 VDU 31,P,8,240,32,32,32,240
 1140 VDU 31,P,9,240,32,32,B,240
 1150 VDU 31,P,10,237,239,239,239,238
```

```
 1160 SOUND 1,-15,100,1
 1170 REPEAT
 1180 COLOUR 3
 1190 PROCwait
 1200 PRINT TAB(0,20)"   I HAVE ";CT
 1210 PRINT TAB(0,22)"  YOU HAVE ";T0
 1220 IF CT>21 THEN PROCbust
 1230 IF CT=21 THEN PROCpontoon
 1240 IF CT>T0 THEN PROCwin

 1250 CC=RND(9)
 1260 CT=CT+CC
 1270 P=P+6
 1280 B=RND(4)+229
 1290 COLOUR 1
 1300 PROCwait
 1310 IF P>16 P=2:CLS:PROCcard
 1320 PRINT TAB(0,1)"THE COMPUTER'S CARD
S"
 1330 PRINT TAB(0,29)"YOU:$";MONEY;TAB(1
0);"COMP:$";CMON
 1340 VDU 31,P,2,235,239,239,239,236
 1350 VDU 31,P,3,240,B,32,32,240
 1360 VDU 31,P,4,240,32,32,32,240
 1370 VDU 31,P,5,240,32,32,32,240
 1380 VDU 31,P,6,240,32,CC+48,32,240
 1390 VDU 31,P,7,240,32,32,32,240
 1400 VDU 31,P,8,240,32,32,32,240
 1410 VDU 31,P,9,240,32,32,B,240
 1420 VDU 31,P,10,237,239,239,239,238

 1430 SOUND 1,-15,100,1
```

```
1440 COLOUR 3
1450 UNTIL FALSE

1460 DEF PROCbust
1470 PROCwait
1480 PRINT TAB(0,20)"       I BUST "
1490 PRINT TAB(0,22)"   YOU WIN $25"
1500 MONEY=MONEY+25
1510 CMON=CMON-25
1520 PRINT TAB(2,24)"YOU HAVE $";MONEY
1530 SOUND 1,-15,150,4
1540 PROCwait
1550 PROCwait
1560 CLS
1570 GOTO 820
1580 ENDPROC

1590 DEF PROCpontoon
1600 PROCwait
1610 PRINT TAB(0,20)"    PONTOON!!"
1620 SOUND 1,-15,0,10
1630 MONEY=MONEY-45
1640 CMON=CMON+45
1650 PRINT TAB(0,22)"  YOU LOSE $45"
1660 PRINT TAB(0,24)"  YOU HAVE $";MONE
Y
1670 PROCwait
1680 PROCwait
1690 PROCcheck
1700 CLS
1710 GOTO 820
1720 ENDPROC
```

```
 1730 DEF PROCwin
 1740 PROCwait
 1750 PRINT TAB(0,20)"     I WIN !!   "
 1760 MONEY=MONEY-35
 1770 CMON=CMON+35
 1780 PRINT TAB(0,22)"  YOU LOSE $35"
 1790 PRINT TAB(0,24)"  YOU HAVE $";MONE
Y
 1800 SOUND 1,-15,50,50
 1810 PROCwait
 1820 PROCwait
 1830 CLS
 1840 PROCcheck
 1850 GOTO 100
 1860 ENDPROC

 1870 DEF PROCpon
 1880 PROCwait
 1890 PRINT TAB(0,20)"   PONTOON!!"
 1900 MONEY=MONEY+50
 1910 CMON=CMON-50
 1920 PRINT TAB(0,22)"  YOU WIN $50"
 1930 PRINT TAB(1,24)"YOU HAVE $";MONEY
 1940 SOUND 1,-15,250,50
 1950 PROCwait
 1960 PROCwait
 1970 CLS
 1980 GOTO 820
 1990 ENDPROC
 2000 DEF PROCcheck
 2010 IF MONEY<0 THEN PROCend
 2020 IF CMON<0 THEN PROCwend
 2030 ENDPROC
```

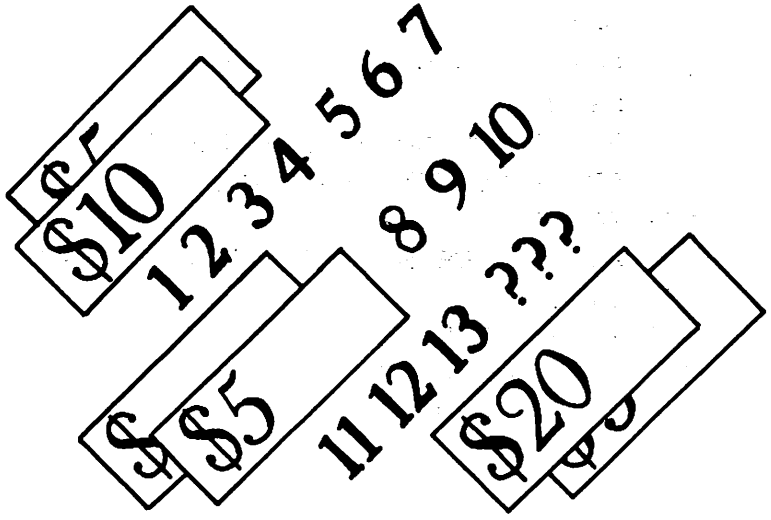
```
 2040 DEF PROCchr

 2050 VDU 23,230,24,60,126,255,255,126,6
0,24
 2060 VDU 23,231,24,60,90,255,255,90,24,
60
 2070 VDU 23,232,16,56,124,254,254,84,16
,56
 2080 VDU 23,233,68,238,254,254,254,124,
56,16
 2090 VDU 23,234,204,204,51,51,204,204,5
1,51
 2100 VDU 23,235,0,0,0,7,15,12,24,24
 2110 VDU 23,236,0,0,0,192,240,48,24,24
 2120 VDU 23,237,24,24,12,15,7,0,0,0
 2130 VDU 23,238,24,24,48,240,192,0,0,0
 2140 VDU 23,239,0,0,0,255,255,0,0,0
 2150 VDU 23,240,24,24,24,24,24,24,24,24
 2160
 2170 ENDPROC
 2180 DEF PROCwait
 2190 FOR T=1 TO 5000
 2200 NEXT
 2210 ENDPROC

 2240 DEF PROCend
 2250 PRINT"YOU RAN OUT OF MONEY";
 2260 A=RND(9)
 2270 SOUND 1,-15,A+RND(3),1
 2280 SOUND 2,-15,A+RND(6),1
 2290 SOUND 3,-15,A+RND(1),1
 2300 GOTO 2250
```

```
 2310 ENDPROC
 2320 DEF PROCcard
 2330 VDU 31,14,15,235,239,239,239,236
 2340 FOR CA=16 TO 22
 2350 VDU 31,14,CA,240,234,234,234,240
 2360 NEXT
 2370 VDU 31,14,23,237,239,239,239,238
 2380 ENDPROC
 2390 DEF PROCwend
 2400 PRINT TAB(0,20)"WELL DONE,YOUR"
 2410 PRINT TAB(0,22)" OPPONENT HAS"
 2420 PRINT TAB(0,24)"LOST HIS MONEY"
 2430 COLOUR 1
 2440 PRINT TAB(0,26)"  ANY KEY TO START
"
 2450 S=GET
 2460 RUN
 2470 ENDPROC
```

TIGHT SQUEEZE

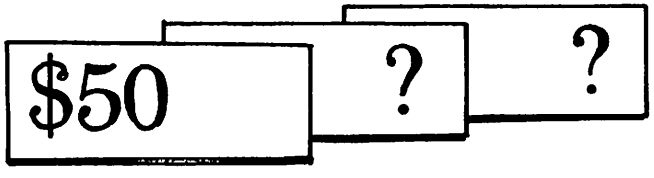
The rules of this game are simple, so it can be fun for everyone. The BBC Micro displays two numbers between 1 and 13 and selects a third, secret number. You start the game with a sum of money in dollars and the computer asks you to place a bet on this secret number being between the two numbers displayed. There is no need to guess the exact number.

If you guess correctly, and the number does lie between those two, you win double the amount you staked. If you are wrong, you lose your stake. If you bet nothing, the Micro will call you 'coward'. The game is over when you run out of money.



```
 10 REM TIGHT SQUEEZE
 20 ON ERROR GOTO 420
 30 MODE 4
 40 VDU 19,0,1,0,0,0
 50 VDU 19,1,0,0,0,0
 60 VDU 28,0,31,20,0
 70 MOVE 640,20
 80 D=20
 90 G=0
100 CLS
110 PRINT
120 A=RND(13)
130 B=RND(13)
140 IF ABS(B-A)<2 THEN 120
150 C=RND(13)
160 IF A=C OR B=C THEN 120
170 PRINT"MY FIRST NUMBER IS ";A
180 PRINT"      MY SECOND IS ";B
190 PRINT
200 PRINT"YOU HAVE $";D;"  "
210 PRINT
220 PRINT"HOW MUCH DO YOU BET"
230 PRINT"MY NEXT NUMBER LIES"
240 PRINT
250 PRINT'" BETWEEN ";A;" AND ";B;" ";
260 INPUT E
270 IF E>D THEN 260
280 IF E<0 THEN PRINT TAB(5);"Coward!"
290 PROCwait(999)
300 PRINT
310 PRINT"MY NUMBER IS ";C
320 IF E<1 THEN 420
```

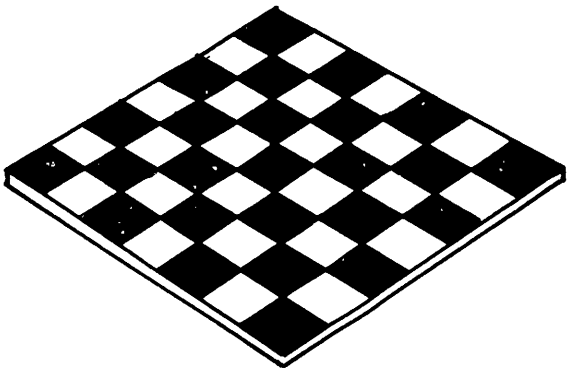
```
 330 IF NOT(C>A AND C<B OR C<A AND C>B)
THEN 380
 340 PRINT
 350 PRINT"WELL DONE! YOU WIN $";2*E
 360 D=D+2*E
 370 GOTO 420
 380 PRINT
 390 PRINT"SORRY, YOU LOSE $";E
 400 D=D-E
 410 IF D<0 THEN 480
 420 PROCwait(3000)
 430 DRAW 672+G*8,D*3
 440 SOUND 19,-15,D+6,5
 450 IF G>78 CLG:G=0:MOVE 672,D+3
 460 G=G+1
 470 GOTO 100
 480 PRINT"THE GAME IS OVER!!"
 490 PRINT"YOU ARE BROKE ";
 500 PROCwait(1000)
 510 GOTO 490
 520 DEF PROCwait(B)
 530 LOCAL A
 540 FOR A=1 TO B
 550 NEXT
 560 ENDPROC
```



DRAUGHTS

This game is played in exactly the same way as conventional draughts, that is on an 8 x 8 board, with 12 pieces for each player. Play is confined to the black squares, and all the moves are diagonal. You can move your pieces up the board only, and the computer can only move down. This continues until pieces reach the far side of the board, where they become Kings; a King can move in either direction. You can only move onto an adjoining square, although if there is one of the computer's pieces on an adjoining square and a vacant square beyond, you can take that piece by jumping over it.

To make a move you must enter the co-ordinates of the square from which you want to move, 4 4 for example, then press 'Return' and enter the co-ordinates of the square you want to move to, 5 5 for example. Co-ordinates are made up of a number from the vertical side of the board followed by a number from the horizontal side of the board, without a separating comma or full stop.



```
   20 REM CHECKERS
   30
   40 VDU 23,224,&00,&3C,&66,&66,&66,&66
,&3C,&00
   50 VDU 23,225,&00,&3C,&7E,&7E,&7E,&7E
,&3C,&00
   60 VDU 23,230,&FF,&FF,&FF,&FF,&FF,&FF
,&FF,&FF
   70 VDU 23,226,&00,&42,&66,&66,&7E,&7E
,&7E,&00
   80 MODE 4

   90 PROCinitialise

  100 REPEAT
  110 PROCprint_board
  120 PROCplayer_move
  130 PROCprint_board
  140 PROCcomputer_move
  150 UNTIL FALSE

  160 DEF PROCcomputer_move
  170 REM COMPUTER CAPTURES, SAFE
  180 FOR M=1 TO 20
  190 S(M)=0
  200 SOUND 1,-15,200-10*M,1
  210 NEXT M
  220 SC=0
  230 A=89
  240 A=A-1
  250 IF Q(A)<>C AND Q(A)<>CK THEN 330
  260 B=0
  270 B=B+1
```

```
  280 M=H+N(B)
  290 IF M>88 OR M<11 THEN 320
  300 IF (Q(M)=H OR Q(M)=HK) AND Q(M+N(B
))=E THEN 370
  310 IF Q(M)=E THEN IF (Q(M-11)<>H AND
Q(M-11)<>HK) THEN IF (Q(M-9)<>H AND Q(M-
9)<>HK) AND Q(M+9)<>HK AND Q(M+11)<>HK T
HEN PROCsafe_moves
  320 IF B<2 OR (Q(A)=CK AND B<4) THEN 2
70
  330 IF A>11 THEN 240
  340 FL=0:IF Q(22)=C OR Q(24)=C OR Q(26
)=C OR Q(28)=C THEN PROCking_advance
  350 IF FL=1 THEN 740
  360 GOTO550
  370 Q(M+N(B))=Q(A)
  380 Q(M)=E
  390 Q(A)=E
  400 CO=CO+1
  410 PROCprint_board
  420 A=M+N(B)
  430 B=0
  440 B=B+1
  450 IF A+2*N(B)<11 OR A+2*N(B)>88 THEN
 440
  460 M=A+N(B)
  470 IF Q(M)=C AND B>3 THEN ENDPROC
  480 IF (Q(M)=H OR Q(M)=HK) AND Q(M+N(B
))=E THEN 370
  490 IF B<2 OR (Q(A)=CK AND B<4) THEN 4
40
  500 ENDPROC
```

```
  510 DEF PROCsafe_moves
  520 IF SC<20 THEN SC=SC+1
  530 S(SC)=100*A+B+20
  540 ENDPROC

  550 IF SC=0 THEN 610
  560 XC=INT(RND(1)*SC)+1
  570 A=INT(S(XC)/100)
  580 M=A+N(S(XC)-100*A-20)
  590 GOTO 740
  600 ENDPROC

  610 SC=SC+1
  620 A=INT(RND(1)*88)+1
  630 IF Q(A)<>C AND Q(A)<>CK THEN 700
  640 B=0
  650 B=B+1
  660 M=A+N(B)
  670 IF M>88 OR M<11 THEN 690
  680 IF Q(M)=E THEN 740
  690 IF B<2 OR Q(A)=CK AND B<4 AND RND(
1)>.6 THEN 650
  700 IF SC<100 THEN 610
  710 PRINT "I CONCEDE THE GAME"
  720 END
  730 REM MAKE MOVE
  740 Q(M)=Q(A)
  750 Q(A)=E:VDU 7
  760 ENDPROC

  770 DEF PROCprint_board
  780 PRINT TAB(3,4)"COMPUTER:";CO;"
```

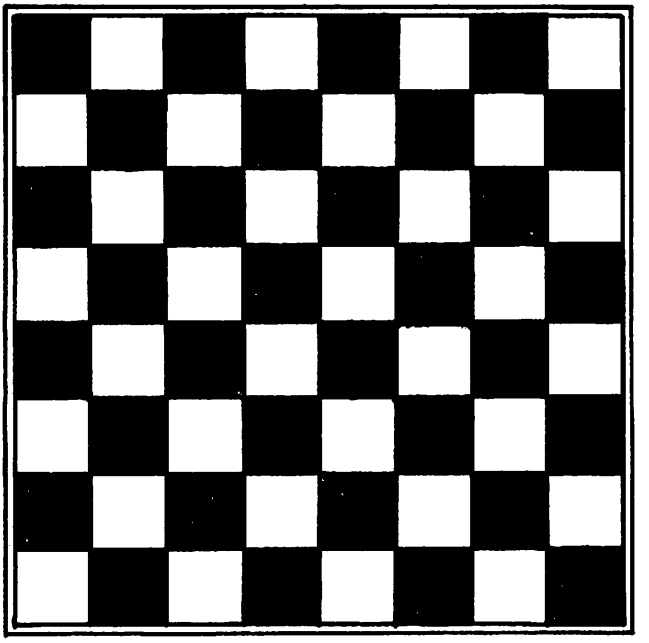
```
        HUMAN:";HU
  790 PRINTTAB(4,8);"1 2 3 4 5 6 7 8 "
  800 PRINT
  810 FOR F=80TO10 STEP-10
  820 PRINTTAB(2);F/10;CHR$(9);
  830 FOR G=1TO8
  840 VDU Q(F+G),9
  850 NEXT G
  860 PRINT;F/10
  870 PRINT
  880 NEXT F
  890 PRINTTAB(4);"1 2 3 4 5 6 7 8 "
  900 IF CO=12 OR HU=12 THEN 920
  910 ENDPROC

  920 IF CO=12 THEN PRINT'"YOU WIN"
  930 IF HU=12 THEN PRINT'"I WIN"
  940 END

  950 REM COMPUTER KINGS, PLAYER MOVE
  960 DEF PROCplayer_move
  970 FOR T=11 TO 17
  980 IF Q(T)=C THEN Q(T)=CK
  990 NEXT
 1000 SOUND 1,-15,250,3
 1010 REM PLAYER ENTERS 99 TO CONCEDE
 1020 VDU 31,23,13
 1030 INPUT "FROM "A
 1040 IF A=99 THEN 1770
 1050 VDU 31,23,16
 1060 SOUND 1,-15,230,3
 1070 INPUT "TO   "B
```

```
 1080 PRINT TAB(23,13)"           "
 1090 PRINT TAB(23,16)"           "
 1100 PRINT TAB(0,0)SPC(40)
 1110 SOUND 1,-15,210,3
 1120 Q(B)=Q(A)
 1130 Q(A)=E
 1140 FOR T=82 TO 88
 1150 IF Q(T)=H THEN Q(T)=HK
 1160 NEXT T
 1170 IF ABS(A-B)<12 THEN ENDPROC
 1180 HU=HU+1
 1190 Q((A+B)/2)=E
 1200 PROCprint_board
 1210 FOR T=82 TO 88
 1220 IF Q(T)=H THEN Q(T)=HK
 1230 NEXT T
 1240 INPUT"CAN YOU JUMP AGAIN (Y OR N)"
A$:VDU 13:PRINT"
   ":PROCprint_board
 1250 IF A$<>"Y" ENDPROC
 1260 A=B
 1270 GOTO 1040

 1280 REM INITIALISE
 1290 DEF PROCinitialise
 1300 CLS:REM CLEAR SCREEN
 1310 PROCsqu
 1320 VDU 23;8202;0;0
 1330 VDU 19,0,4,0,0,0
 1340 H=224:REM HUMAN PIECE
 1350 HK=75:REM HUMAN KING
 1360 C=225:REM COMPUTER PIECE
```

```
 1370 CK=226:REM COMPUTER KING
 1380 E=32:REM EMPTY SQUARE
 1390 B=9:REM BLACK SQUARE
 1400 OF=-99:REM OFF BOARD
 1410 DIM Q(99): REM BOARD
 1420 DIM N(4):REM MOVES BY COMPUTER
 1430 DIM S(20):REM SAFE MOVES
 1440 FOR M=1 TO 99
 1450 Q(M)=OF
 1460 NEXT M
 1470 FOR M=1 TO 64
 1480 READ D:READ G
 1490 Q(D)=G
 1500 NEXT M
 1510 DATA 81,B,82,C,83,B,84,C,85,B,86,C
,87,B
 1520 DATA 88,C,71,C,72,B,73,C,74,B,75,C
,76,B
 1530 DATA 77,C,78,B,61,B,62,C,63,B,64,C
 1540 DATA 65,B,66,C,67,B,68,C,51,E,52,B
 1550 DATA 53,E,54,B,55,E,56,B,57,E,58,B
 1560 DATA 41,B;42,E,43,B,44,E,45,B,46,E
 1570 DATA 47,B,48,E,31,H,32,B,33,H,34,B
,35,H
 1580 DATA 36,B,37,H,38,B,21,B,22,H,23,B
,24,H
 1590 DATA 25,B,26,H,27,B,28,H,11,H,12,B
,13,H
 1600 DATA 14,B,15,H,16,B,17,H,18,B
 1610 FOR M=1 TO 4
 1620 READ N(M)
 1630 NEXT M
```

```
 1640 DATA -11,-9,11,9
 1650 CO=0
 1660 HU=0
 1670 ENDPROC

 1680 DEF PROCking_advance
 1690 IF Q(22)=C AND Q(11)=E THEN A=22:M
=11:FL=1:ENDPROC
 1700 IF Q(22)=C AND Q(13)=E THEN A=22:M
=13:FL=1:ENDPROC
 1710 IF Q(24)=C AND Q(13)=E THEN A=24:M
=13:FL=1:ENDPROC
 1720 IF Q(24)=C AND Q(15)=E THEN A=24:M
=15:FL=1:ENDPROC
 1730 IF Q(26)=C AND Q(15)=E THEN A=26:M
=15:FL=1:ENDPROC
 1740 IF Q(26)=C AND Q(17)=E THEN A=26:M
=17:FL=1:ENDPROC
 1750 IF Q(28)=C AND Q(17)=E THEN A=28:M
=17:FL=1:ENDPROC
 1760 ENDPROC
 1770 PRINT "I ACCEPT YOUR WISH  TO CONC
EDE"
 1780 PRINT'"THANKS FOR THE GAME"
 1790 END

 1800 DEF PROCsqu
 1810 GCOL 0,129
 1820 FOR Y=208 TO 592 STEP 128
 1830 VDU 24,112;Y;624;Y+64;
 1840 CLG
 1850 NEXT
```

```
1860 GCOL 4,129
1870 FOR X=112 TO 496 STEP 128
1880 VDU 24,X;208;X+64;720;
1890 CLG
1900 NEXT
1910 VDU 26
1920 MOVE 112,208
1930 DRAW 112,720
1940 DRAW 624,720
1950 DRAW 624,208
1960 DRAW 112,208
1970 ENDPROC
```



LIFE CHANGE UNITS

This program evaluates recent changes in your life in terms of the effect those changes have had on you. Any change in your life, good or bad, creates measurable stress. A number of situations will be printed in the form of a question on the screen. If the situation mentioned is part of your present life, if, for example, you have started a new job, press the "Y" key. If the answer is no, press "N". At the end of the questioning, an assessment will be made.

```
10 REM LIFE CHANGE UNITS

20 MODE 7
30 PROCintro
40 PROCquestions
50 PROCanswer
60 END
70 DEF PROCintro
80 L=0
90 PRINT'SPC(11);"L. C. U. TEST"
100 PRINT'" HOW CHANGES IN YOUR LIFEST
YLE EFFECT";SPC(15);"YOUR FUTURE"
110 PRINT"Answer Y if you think that t
he state-   ment is applicable to you,ov
er the last 3 months or over the next 3
months.     Answer N if you think that t
he state-   ment is not applicable to yo
u."
120 ENDPROC

130 DEF PROCtest
140 PRINT TAB(35);L
150 PRINT'CHR$(RND(6)+128);L$
160 REPEAT
170 A$=GET$
180 UNTIL A$="Y" OR A$="N"
190 IF A$="Y" L=L+A
200 SOUND 17,-15,L,4
210 SOUND 18,-15,L+3,4
220 SOUND 19,-15,L+6,4
230 ENDPROC
```

```
240 DEF PROCquestions
250 REM THIS QUESTION WORTH 8
260 L$="Death of spouse"
270 A=8
280 PROCtest

290 REM THIS QUESTION WORTH 7
300 L$="Divorce"
310 A=7
320 PROCtest

330 REM THIS QUESTION WORTH 6
340 L$="Giving up hard drugs"
350 A=6
360 PROCtest

370 REM THIS QUESTION WORTH 6
380 L$="Marital separation"
390 A=6
400 PROCtest

410 REM THIS QUESTION WORTH 6
420 L$="Prison sentance"
430 A=6
440 PROCtest

450 CLS
460 REM THIS QUESTION WORTH 7
470 L$="Death of a close relation"
480 A=7
490 PROCtest
```

```
500 REM THIS QUESTION WORTH 4
510 L$="Personal injury or illness"
520 A=4
530 PROCtest

540 REM THIS QUESTION WORTH 4
550 L$="Marriage"
560 A=4
570 PROCtest

580 REM THIS QUESTION WORTH 5
590 L$="Sacked or made redundant"
600 A=5
610 PROCtest

620 REM THIS QUESTION WORTH 1
630 L$="Marital concilation"
640 A=1
650 PROCtest

660 REM THIS QUESTION WORTH 2
670 L$="Retirement"
680 A=2
690 PROCtest

700 REM THIS QUESTION WORTH 3
710 L$="Change in relations health"
720 A=3
730 PROCtest

740 CLS
750 REM THIS QUESTION WORTH 3
```

```
  760 L$="Pregnacy"
  770 A=3
  780 PROCtest

  790 REM THIS QUESTION WORTH 6
  800 L$="Giving up >40 cigarettes a day
"
  810 A=6
  820 PROCtest

  830 REM THIS QUESTION WORTH 5
  840 L$="Sex difficulties"
  850 A=5
  860 PROCtest

  870 REM THIS QUESTION WORTH 2
  880 L$="Increase in members of family"
  890 A=2
  900 PROCtest

  910 REM THIS QUESTION WORTH 3
  920 L$="Business readjustment"
  930 A=3
  940 PROCtest

  950 REM THIS QUESTION WORTH 2
  960 L$="Change in financial state"
  970 A=2
  980 PROCtest

  990 REM THIS QUESTION WORTH 8
 1000 L$="Death of a close friend"
```

```
 1010 A=8
 1020 PROCtest

 1030 REM THIS QUESTION WORTH 2
 1040 L$="Change to a different line of
work"
 1050 A=2
 1060 PROCtest

 1070 CLS
 1080 REM THIS QUESTION WORTH 3
 1090 L$="More or less arguments with sp
ouse"
 1100 A=3
 1110 PROCtest

 1120 REM THIS QUESTION WORTH 3
 1130 L$="Pre-menstural tension"
 1140 A=3
 1150 PROCtest

 1160 REM THIS QUESTION WORTH 7
 1170 L$="Foreclosure of mortgage or loa
n"
 1180 A=7
 1190 PROCtest

 1200 REM THIS QUESTION WORTH 3
 1210 L$="Mortgage over 30,000 pounds"
 1220 A=3
 1230 PROCtest
```

```
 1240 REM THIS QUESTION WORTH 2
 1250 L$="Change in work responsbility"
 1260 A=2
 1270 PROCtest

 1280 REM THIS QUESTION WORTH 2
 1290 L$="Trouble with in-laws"
 1300 A=2
 1310 PROCtest

 1320 REM THIS QUESTION WORTH 1
 1330 L$="Outstanding personal achievmen
t"
 1340 A=1
 1350 PROCtest

 1360 REM THIS QUESTION WORTH 1
 1370 L$="Spouse starts or stops work"
 1380 A=1
 1390 PROCtest

 1400 CLS
 1410 REM THIS QUESTION WORTH 2
 1420 L$="Change in living conditions"
 1430 A=1
 1440 PROCtest

 1450 REM THIS QUESTION WORTH 4
 1460 L$="Change in personal habits"
 1470 A=4
 1480 PROCtest
```

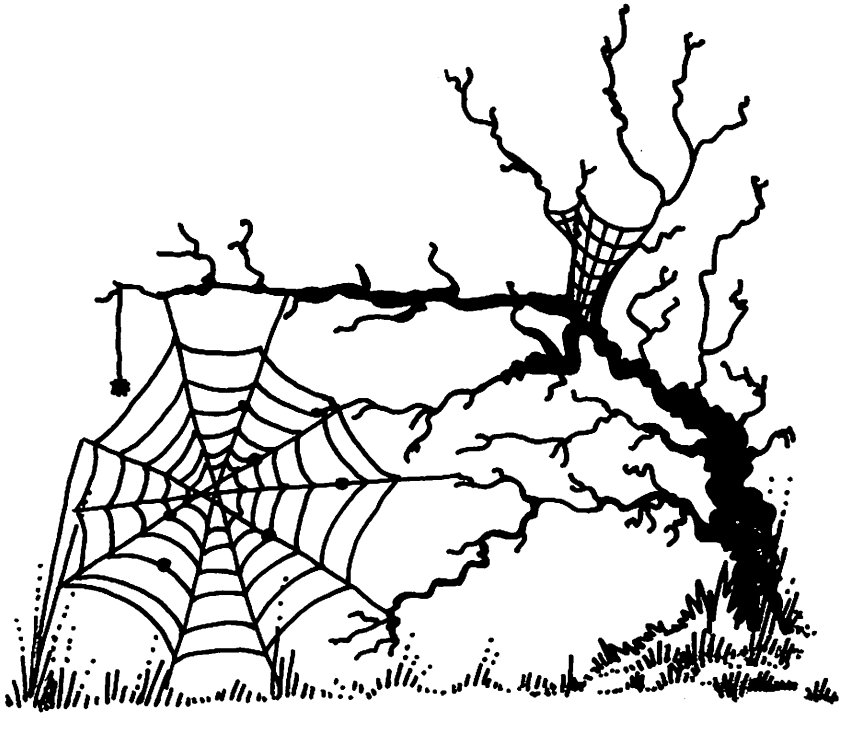
```
 1490 REM THIS QUESTION WORTH 3
 1500 L$="Trouble with your boss"
 1510 A=3
 1520 PROCtest

 1530 REM THIS QUESTION WORTH 5
 1540 L$="Giving up <20 cigarettes a day
"
 1550 A=5
 1560 PROCtest

 1570 REM THIS QUESTION WORTH 2
 1580 L$="Change in work hours or condit
ions"
 1590 A=2
 1600 PROCtest

 1610 REM THIS QUESTION WORTH 2
 1620 L$="Change in residence"
 1630 A=2
 1640 PROCtest

 1650 REM THIS QUESTION WORTH 2
 1660 L$="Change of school"
 1670 A=2
 1680 PROCtest

 1690 REM THIS QUESTION WORTH 1
 1700 L$="Change in social activity"
 1710 A=1
 1720 PROCtest
```

```
 1730 CLS
 1740 REM THIS QUESTION WORTH 3
 1750 L$="Mortgage under 20,000 pounds"
 1760 A=3
 1770 PROCtest

 1780 REM THIS QUESTION WORTH 2
 1790 L$="Change in sleeping habits"
 1800 A=2
 1810 PROCtest

 1820 REM THIS QUESTION WORTH 1
 1830L$="Change in No. of family get-tog
ethers"
 1840 A=1
 1850 PROCtest

 1860 REM THIS QUESTION WORTH 1
 1870 L$="Change in eating habits"
 1880 A=1
 1890 PROCtest

 1900 REM THIS QUESTION WORTH 2
 1910 L$="Going on holiday"
 1920 A=2
 1930 PROCtest

 1940 REM THIS QUESTION WORTH 1
 1950 L$="It is christmas"
 1960 A=1
 1970 PROCtest

 1980 REM THIS QUESTION WORTH 3
 1990 L$="Minor violation of the law"
 2000 A=3
```

```
 2010 PROCtest

 2020 ENDPROC

 2030 DEF PROCanswer
 2040 CLS
 2050 PRINT''" YOUR LIFE CHANGE UNITS T
OTAL IS ";L
 2060 PRINT''" THE VERDICT. ."
 2070 FOR T=1 TO 5000:NEXT
 2080 IF L<50 PRINT''"You are living a s
table,safe way of lifeand you are less l
ikely to have an acci-dent or become ill
 "
 2090 IF L<19 PRINT'"I SUSPECT THAT YOU
MAY RECEIVE INVATAT- IONS TO JOIN A NUNN
ERY , OR IF YOU ARE  MALE TO A MONESTRY!
!"
 2100 IF L>49 AND L<75 PRINT''"You are d
oing O.K. There is only a 37%  chance of
 you encountering health or sa-fety prob
lems."
 2110 IF L>74 AND L<109 PRINT''"You had
better slow down ! You have a  51% chan
ce that you will encounter heal-th safet
y problems."
 2120 IF L>110 PRINT''CHR$(129);"KEEP AW
AY FROM SLEEPING PILLS , RAZORS ";CHR$(1
29);"AND MEMORIZE THE SAMARITANS";CHR$(1
29);"'PHONE NO. !!!!"
 2130 ENDPROC
```

SPIDER AND FLY

You are a very hungry spider, hanging from a thread on the left-hand side of the screen, whose main aim in life is to get your lunch. When six rather fat flies come into view, you have 99 seconds to munch your way through as many of them as you can. Use the "Y" key to go up and the "B" key to go down.



```
 20 REM SPIDER N' FLY
 30
 40 HI=0
 50 PROCchr

 60 MODE 4
 70 PROCinit
 80 PRINT TAB(0,1);"SCORE ";SC

 90 A%=INKEY(0)
100 *FX 11,1
110 *FX 12,1
120 PRINT TAB(0,V);"  "
130 IF A%=89 V=V-1
140 IF A%=66 V=V+1
150 IF V>30 V=V-2
160 IF V<3 V=V+2
170 *FX 15

180 SOUND 17,-9,-(V*7),10
190 SOUND 18,-9,-(V*7+1),10
200 SOUND 19,-9,-(V*7+2),10
210 IF FNread(2,V)<>32 THEN PROCgulp
220 PRINT TAB(0,V-1)"|"
230 VDU 31,0,V,230,231

240 FOR A=1 TO 9
250 VDU 31,F(A),A*3,32,32
260 F(A)=F(A)-(RND(16)/10)
270 IF F(A)<2 PRINT TAB(1,A*3)" ":F(A)
=39:FOR AA=2 TO V-1:PRINT TAB(0,AA);"|":
NEXT
```

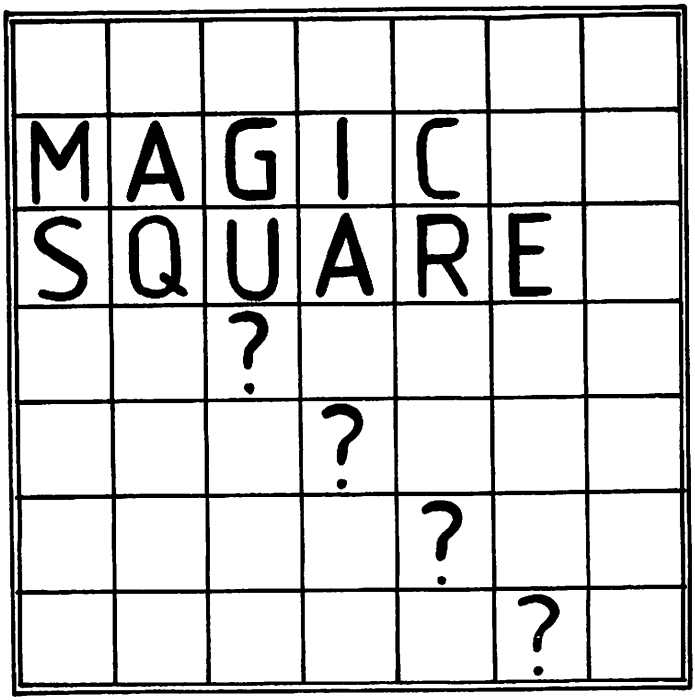
```
280 VDU 31,F(A),A*3,232,233
290 NEXT
300 PRINT TAB(19,1) "Time :";TIME/100

310 IF TIME>10000 PROCend
320 GOTO 90

330 DEF PROCgulp
340 SC=SC+1
350 SOUND 16,-10,200,2
360 SOUND 16,-10,180,2
370 SOUND 16,-10,200,2
380 PRINT TAB(2,V);"   "
390 PRINT TAB(0,1);"SCORE ";SC
400 ENDPROC

410 DEF PROCend
420 PRINT TAB(13,12)"GAME OVER!!"
430 PRINT TAB(10,14)"WITH A SCORE OF "
;SC
440 IF SC>HI THEN HI=SC
450 PRINT TAB(11,16)"HIGH-SCORE :";HI
460 PRINT TAB(0,24)"PRESS ANY KEY TO S
TART AGAIN"
470 FOR T=1 TO 9000:NEXT
480 A=GET
490 RUN

500 DEF PROCinit
510 VDU 23;8202;0;0;0;
520 VDU 19,0,4,0,0,0
530 V=10
```

```
  540 DIM F(20)
  550 FOR A=1 TO 6
  560 F(A)=28
  570 NEXT
  580 SC=0
  590 FOR A=2 TO V-1
  600 PRINT TAB(0,A)"|"
  610 NEXT
  620 TIME=0
  630 ENDPROC

  640 DEF PROCchr
  650 VDU 23,230,60,126,255,255,127,127,
149,148
  660 VDU 23,231,0,108,248,220,252,248,9
6,0
  670 VDU 23,232,0,17,63,94,111,23,45,0
  680 VDU 23,233,0,252,2,2,252,224,0,0
  690 ENDPROC

  700 DEF FNread(X%,Y%)
  710 LOCAL A%,C%
  720 A%=135
  730 VDU 31,X%,Y%
  740 C%=((USR(&FFF4) AND &FFFF) DIV &10
0)
  750 IF C%<128 THEN=C%
  760=(C% MOD 32)+224
```

MAGIC SQUARE

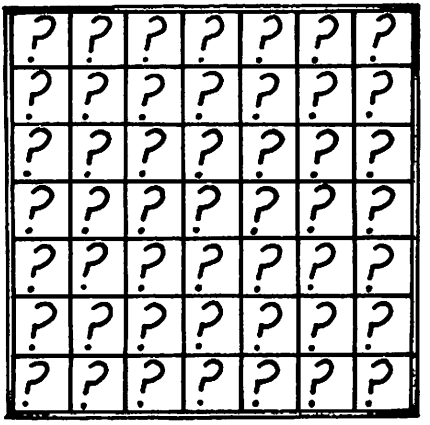
This program will generate four different sizes of magic squares using the de la Loubere method. A 'magic square' is a square of numbers where all the rows, columns, and leading diagonals individually total the same number.

When you run the program, you'll be asked to enter an integer between one and nine. The screen then clears, and a magic square is constructed. The computer then deletes some of the numbers in the magic square and asks you what the missing numbers are.



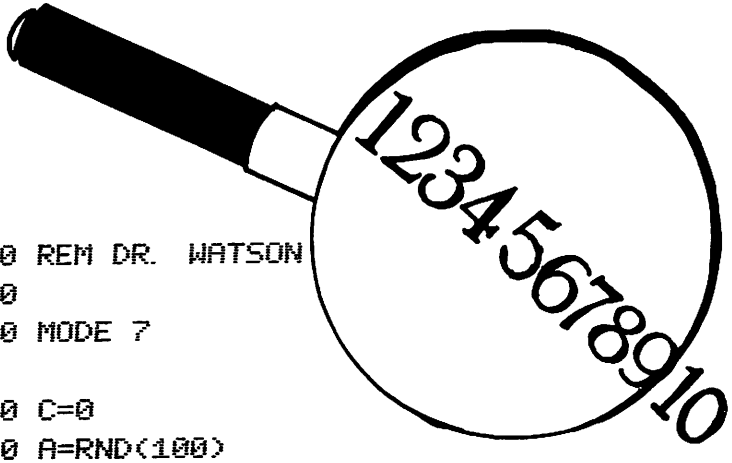
```
 20 REM MAGIC SQUARE
 30
 40 MODE 7
 50 DIM M(47,47)
 60 DIM R(12)
 70 DIM Q(12)
 80 INPUT'"NO. OF ROWS/COLUMNS",N
 90 CLS
100 PRINT "MAGIC NO. :"
110 PRINT "ROWS/COLS:  ";N
120 SC=0
130 C1=0
140 C=INT(N/2)+1
150 R=1
160 C1=C1+1
170 M(R,C)=C1
180 PRINT TAB(C*3,R*2+1);C1
190 D=N*N
200 IF C1=D THEN 330
210 IF C1/N<>INT(C1/N) THEN 240
220 R=R+1
230 GOTO 160
240 C=C+1
250 IF C<=N THEN 290
260 C=1
270 R=R-1
280 GOTO 160
290 R=R-1
300 IF R>0 THEN GOTO 160
310 R=N
320 GOTO 160
330 T=0
```

```
  340 FOR I=1 TO N
  350 T=T+M(I,1)
  360 NEXT
  370 PRINT TAB(12,0);T
  380 FOR D=1 TO N+2
  390 C=RND(N)
  400 R=RND(N)
  410 IF M(R,C)>1000 THEN GOTO 390
  420 M(R,C)=M(R,C)+1000
  430 Q(D)=M(R,C)-1000
  440 PRINT TAB(C*3,R*2+1);CHR$(8);CHR$(
129);D;CHR$(135)
  450 NEXT
  460 VDU 28,0,24,39,20
  470 FOR D=1 TO N+2
  480 PRINT'"WHAT IS THE NUMBER AT ";CHR
$(129);D
  490 INPUT R(D)
  500 IF R(D)=Q(D):SC=SC+1
  510 PRINT'CHR$(131);"SCORE:";SC
  520 NEXT
```



DR WATSON

This is a game in which the BBC Micro thinks of a number between 1 and 100 and challenges you to guess it. When you make a guess, the computer tells you whether your guess is too high or too low. When you hit on the correct number, it will tell you how many guesses you took. This program runs in Mode 7 and uses Teletext colours.




```
  20 REM DR. WATSON
  30
  40 MODE 7

  50 C=0
  60 A=RND(100)
  70 PRINT
  80 PRINT "I AM THINKING OF A NUMBER B
ETWEEN"
  90 PRINT "ONE AND 100"
 100 PRINT
 110 PRINT "WHAT IS IT?"
 120 PRINT
```

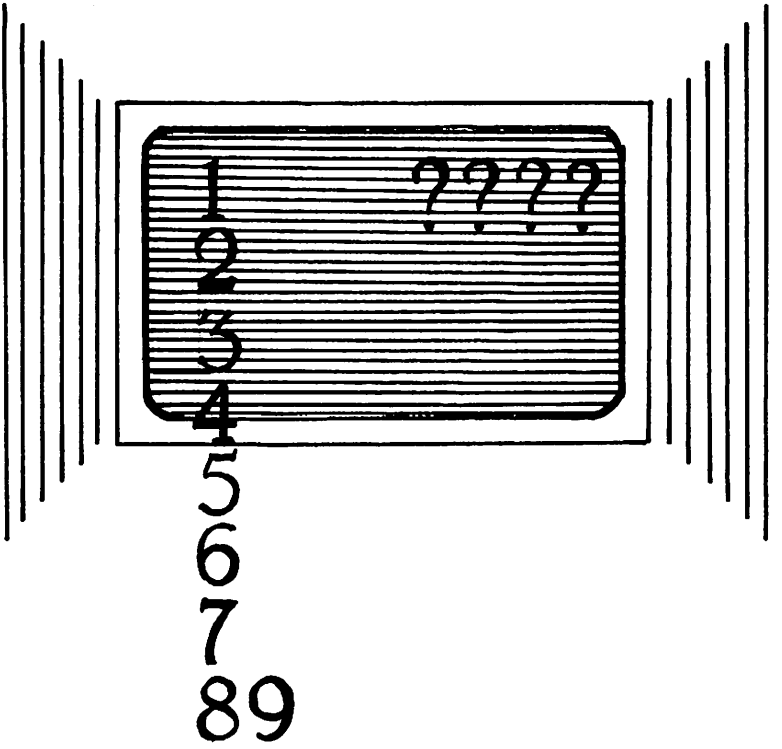
```
  130 INPUT B
  140 C=C+1
  150 REM "WRONG" IN RED
  160 IF B<>A THEN PRINT "NO, ";B;" IS";
CHR$(129);"WRONG";CHR$(135)
  170 IF B=A THEN GOTO 300
  180 PRINT"GUESS ";C;" IS ";
  190 REM "TOO LOW" IN GREEN
  200 IF B<A THEN PRINT CHR$(130);"TOO L
OW"
  210 REM "TOO HIGH" IN PURPLE
  220 IF B>A THEN PRINT CHR$(133);"TOO H
IGH"
  230 PRINT CHR$(135)
  240 IF C<10 THEN GOTO 130

  250 PRINT
  260 PRINT" END OF GAME"
  270 PRINT" I WAS THINKING OF ";A
  280 GOTO 350

  290 PRINT
  300 PRINT" YES,YOU ARE RIGHT!"
  310 PRINT
  320 PRINT" I WAS THINKING OF ";A
  330 PRINT
  340 PRINT" IT TOOK YOU ";C;" GUESSES"
  350 FOR J=1 TO 100000
  360 NEXT
  370 RUN
```

DR AUDIO

This game is a variation of Doctor Watson in which, when you guess the computer's number, the feedback is given as a tone. The pitch of this note varies according to how far your guess is from the correct number. The sounds are programmed in line 130.

```
   10 REM DR. AUDIO
   20 C=0
   30 PRINT "I AM THINKING OF A NUMBER B
ETWEEN"
   40 PRINT "ONE AND 100"
   50 A=RND(100)
   60 PRINT "WHAT IS IT";
   70 INPUT B
   80 C=C+1
   90 REM 'WRONG' IN RED
  100 IF B<>A THEN PRINT "NO. ";B;" IS "
;CHR$(129);"WRONG";CHR$(135)
  110 IF B=A THEN 180
  120 PRINT "GUESS ";C;" IS ";
  130 SOUND 1,-15,ABS(A-B),5
  140 FOR T=1 TO 700:NEXT
  150 IF C<30 THEN 70
  160 PRINT:PRINT "END OF GAME"
  170 PRINT "I WAS THINKING OF ";A: GOTO
 230
```

```
180 PRINT "YES, YOU ARE RIGHT!"
190 PRINT
200 PRINT "I WAS THINKING OF ";A
210 PRINT
220 PRINT "IT TOOK YOU ";C;"GUESSES"
230 FOR J=1 TO 100000 : NEXT
240 RUN
```



NIM

Nim is a traditional game which has many versions. The object of the game, however, is always the same — namely, not to be forced to take the last turn. In this version, the random number generator has baked an ovenful of scones which you and your BBC Micro settle down to scoff ... A limit is put on the number you can take out at a time and the loser is the one who goes last, that is, who eats the last scone.

Although it is easy to write a program that will ensure that the computer never loses, such a program tends to make your BBC Micro an insufferable bore, so this one has built-in fallibility; even so, it will still win more than half the games it plays. (The Achilles Heel of this program is to be found in line 300; this introduces a random element into the program's winning strategy.)

```
  20 REM NIM
  30
  40 MODE 7
  50 PRINT
  60 E=0
  70 F=0
  80 M=0
  90 L=0
 100 Z=RND(20)+10
 110 H=RND(3)+3
 120 PRINT
 130 IF M=0 THEN PRINT "SCONES LEFT:";Z
:PRINT"THE MOST YOU CAN TAKE IS ";H
 140 IF E>0 AND M=0 THEN PRINT "YOU TOO
K ";E;"  I TOOK ";Q
```

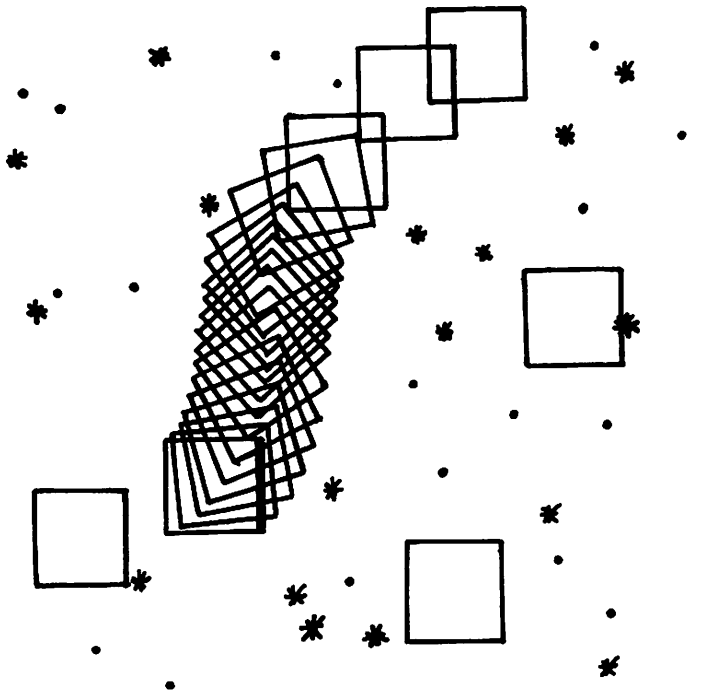
```
  150 PRINT
  160 FOR K=1 TO Z
  170 PRINT"* ";
  180 IF RND(1)<.35 THEN PRINT
  190 NEXT
  200 PRINT
  210 PRINT
  220 IF M=1 THEN PRINT TAB(8);"YOU WIN"
  230 IF M=2 THEN PRINT TAB(12);"I WIN"
  240 IF M>0 THEN GOTO 220
  250 PRINT
  260 PRINT"HOW MANY WILL ";
  270 INPUT"YOU TAKE ",E
  280 IF E<1 OR E>H THEN GOTO 260
  290 Z=Z-E
  300 IF Z>0 THEN GOTO 330
  310 M=2
  320 GOTO 400
  330 Q=Z-1-INT((Z-1)/(H+1))*(H+1)
  340 IF Q=0 THEN Q=RND(H)
  350 IF Q>Z THEN GOTO 330
  360 IF Z>E+2 AND RND(1)<.5 THEN Q=Q+IN
T(RND(1)-.5)
  370 IF Q>H OR Q<1 THEN GOTO 330
  380 Z=Z-Q
  390 IF Z=0 THEN M=1
  400 CLS
  410 GOTO 120
```

CHAIN

This program gives the impression of a long chain 'falling' through space. This is done by drawing and redrawing a four-sided shape on the screen as it scrolls. While the screen scrolls, full stops and asterisks are printed on it, so that it looks as if stars and planets are travelling past and that the chain is revolving and falling through space.

```
 10 REM
          ---------CHAIN---------

 40
 50 MODE 4: REM MODE 0
 60 PROCInitialise
 70 PROCLoop
 80 END

100 DEF PROCInitialise
110 VDU 23;8202;0;0;0;
120 VDU 29,640;300;
130 A%=50:B%=25
140 PRINT TAB(0,20)
150 ENDPROC

170 DEF PROCLoop
180 REPEAT
190   A%=A%+5
200   FOR C%=-A% TO A% STEP RND(A%/B%)
210     GCOL 4,1
```

```
220      DRAW 0,A%/3
230      DRAW C%,0
240      DRAW 0,-A%
250      DRAW -C%,0
270      PROCPrint
280      NEXT C%
290    UNTIL FALSE
300 ENDPROC

320 DEF PROCPrint
330 IF RND(10)>8 A$="*" ELSE A$="."
335 PRINT TAB(RND(79)) A$:REM RND(39)
340 ENDPROC
```



GCOL

This program shows the effect of the GCOL statement on the way lines are drawn. In the BBC Micro User Manual only five kinds of GCOL statements are shown. But this program shows the first 20, and how they affect the way the moire patterns appear.

```
 10 FOR W=10 TO 100 STEP 10
 20 MODE 1
 30 VDU 19,0,4,0,0,0
 40 VDU 29,640;512;
 50 FOR B=50 TO 500 STEP 42
 60 F=RND(W)+1
 70 GCOL F,B/F
 80 FOR A=1 TO B*2 STEP 12
 90 PROCA
100 DRAW B-A,-B
110 PROCA
120 DRAW -B,B-A
130 PROCA
140 DRAW B-A,B
150 PROCA
160 DRAW B,B-A
170 NEXT
180 NEXT
190 NEXT
200 RUN
205 DEF PROCA
210 MOVE 0,0
220 ENDPROC
```

SOUND

This program is designed to show how, even though the sound statement may be simple, it can be used to make startling sound effects when many sound statements are used together. The program shows random sounds, which are given numbers, and defined sounds, which you put the number in, and the computer makes the sound.

```
  10 ON ERROR PROCend_sound
  20

  30 MODE 7
  40 VDU 23;8202;0;0;0;

  50 REPEAT
  60 PROCintro
  70 UNTIL FALSE

  80 DEFPROCintro
  90 CLS
 100 VDU 10,10,132,157,13,10,132,157,13
,10,132,157,13,10,132,157,13,10,132,157,
31,13,4,130,157,135,141,211,207,213,206,
196,160,160,132,157,31,13,5,130,157,135,
141,211,207,213,206,196,160,160,132,157
 110 VDU13,10
 120 PRINT TAB(15,22)"PRESS ESCAPE TO E
ND SOUND"
 130 PRINT TAB(14,24)"PRESS BREAK TO EN
D PROGRAM"
```

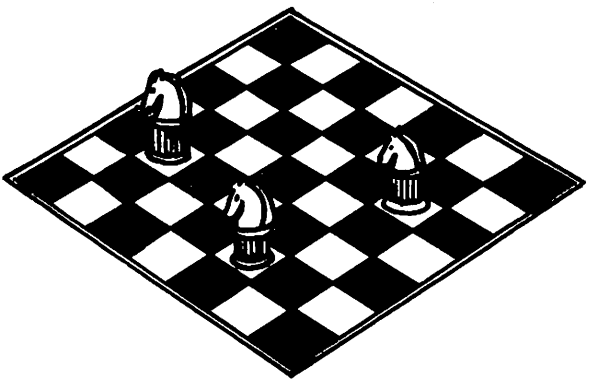
```
  140 PRINT TAB(4,6) "This program is to
 show the versa-"
  150 PRINT TAB(2) "tility of the SOUND
statement";
  160 INPUT ". You can   either have a Ra
ndom sound or a sound"
  170 PRINT "   which is Defined. Key";C
HR$(129);"D";CHR$(135);"for a defined
sound";
  180 PRINT" or key";CHR$(129);"R";CHR$(
135);"for a random sound ";
  190 PRINT "and   to see how it works.
                          THEN KEY RETURN."
  200 REPEAT
  210 C$=GET$
  220 UNTIL C$="R" OR C$="D"
  230 IF C$="R" THEN PROCrnd
  240 IF C$="D" THEN PROCdef
  250 RUN
  260 ENDPROC

  270 DEFPROCrnd
  280 B=RND(50)
  290 A=0
  300 PROCsound
  310 ENDPROC

  320 DEF PROCdef
  330 INPUT TAB(1,15)"   INPUT NUMBER "
B
  340 A=0
  350 PRINT TAB(1,15)"
      "
```

```
  360 PROCsound
  370 ENDPROC

  380 DEFPROCsound
  390 PRINT TAB(20,15);CHR$(128+RND(6));
B;"    "
  400 FORS=1 TO 10 STEP A
  410 FORA=1 TO 255 STEP B
  420 SOUND 17,-RND(15),A,1
  430 SOUND 18,-RND(15),200-A,1
  440 NEXT
  450 NEXT
  460 ENDPROC

  470 DEF PROCend_sound
  480 REPEAT
  490 PRINT TAB(1,19)"Def or Rnd ?"
  500 C$=GET$
  510 UNTIL C$="R" OR C$="D"
  520 PRINT TAB(1,19)"                "
  530 IF C$="R" THEN PROCrnd
  540 IF C$="D" THEN PROCdef
  550 GOTO 490
```

KNIGHTS-BRIDGE

Knightsbridge is a game played on a 7 x 7 draughts board, with each player having seven chess knights as pieces (this explains the name). The computer throws a dice that determines which piece the human player must move, then it throws a dice for itself to determine which piece it must move. All pieces move like knights in chess (and you are not allowed to cheat) and a piece is captured by the opponent landing on top of it. The first player to capture five of the opponent's pieces wins.

You move the indicated piece by telling the computer which square you wish to move to. The board is labelled one to seven along the top and bottom, and from seven to one down the sides, as you can see from the sample run. You enter the number along the side (6 for example) then the number along the top (4 for example) as a single number (that is, 64) to move your piece onto square number 64. You will be swiftly chastised if you try to cheat. You can concede at any time by entering 99 when asked for your move.



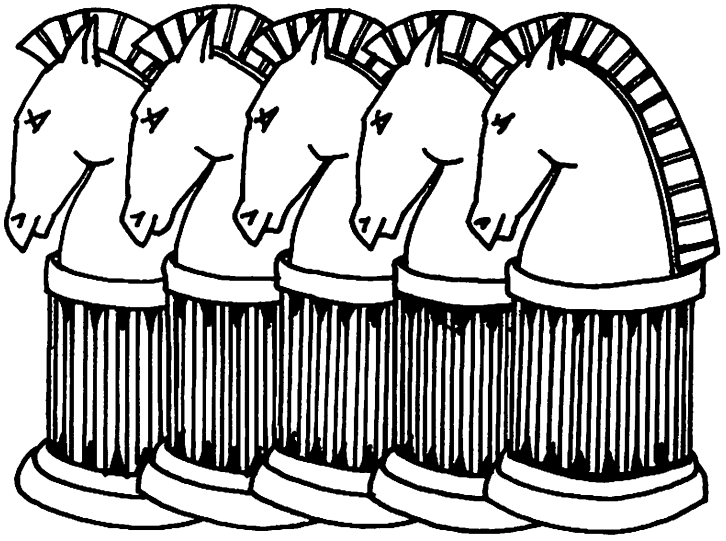
```
   10
   20 REM KNIGHTSBRIDGE
   30
   40 PROCinit
   50 REPEAT
   60 PROCprint_board
   70 PROCcomputer_moves
   80 PROCprint_board
   90 PROCplayer_moves
  100 UNTIL FALSE
  110 END

  120 DEF PROCplayer_moves
  130 Q=0
  140 M=RND(66)+10
  150 Q=Q+1
  160 IF Q=500 THEN PROCend_of_game
  170 IF H(M)<>72 THEN 140
  180 WAIT=TIME
  190 REPEAT UNTIL TIME-WAIT=350
  200 PRINT CHR$(134);"  You must move t
he piece on";CHR$(131);M;"     "
  210 INPUT N
  220 P=0
  230 IF N=99 THEN Q=500 : PROCend_of_ga
me
  240 CT=1
  250 IF M+Z(CT)=N P=1
  260 IF CT<8 CT=CT+1 : GOTO 250
  270 IF P=0 THEN PROCraspberry : GOTO 2
10
  280 IF H(N)=67 HU=HU+1 : PROCjoy
  290 H(M)=255
```

```
  300 H(N)=72
  310 ENDPROC


  320 DEF PROCcomputer_moves
  330 Q1=0
  340 Q1=Q1+1
  350 K=RND(66)+10
  360 IF Q1=500 THEN 620
  370 IF H(K)<>67 THEN 340
  380 WAIT=TIME
  390 REPEAT UNTIL TIME-WAIT=600
  400 PRINT CHR$(133);"I have to move th
e piece on";CHR$(129);K;" "'"    "
  410 WAIT=TIME
  420 REPEAT UNTIL TIME-WAIT=600
  430 W=1
  440 IF K+Z(W)<11 OR K+Z(W)>77 THEN 460
  450 IF H(K+Z(W))=72 THEN PROCvictory :
 GOTO 510
  460 IF W<8 W=W+1 : GOTO 440
  470 W=1
  480 IF (K+Z(W)<11 OR K+Z(W)>77) AND W<
8 W=W+1 : GOTO 480
  490 IF H(K+Z(W))<>255 AND W<8 W=W+1 :
GOTO 490
  500 IF W=8 AND H(K+Z(W))<>255 THEN Q1=
500 : PROCend_of_game
  510 X=K
  520 Y=K+Z(W)
  530 H(X)=255
  540 H(Y)=67
  550 ENDPROC
```

```
  560 DEF PROCvictory
  570 CO=CO+1
  580 FOR Y=1 TO RND(20)+10
  590 SOUND 3,-15,RND(250),4
  600 NEXT
  610 ENDPROC

  620 DEF PROCprint_board
  630 PRINT CHR$(30)'
  640 PROCscore
  650 PRINT TAB(8) "1 2 3 4 5 6 7"
  660 PRINT TAB(7) CHR$(133);"_-_-_-_-_-
_-_"
  670 FOR M=70 TO 10 STEP -10
  680 PRINT TAB(5) CHR$(135);M/10;
  690 FOR N=1 TO 7
  700 IF H(M+N)=255 VDU 130,255
  710 IF H(M+N)=67  VDU 129,42
  720 IF H(M+N)=72  VDU 131,37
  730 NEXT
  740 PRINT CHR$(135);M/10'
  750 NEXT
  760 PRINT TAB(7) CHR$(133);"_-_-_-_-_-
_-_"
  770 PRINT TAB(8) "1 2 3 4 5 6 7"
  780 ENDPROC

  790 DEF PROCend_of_game
  800 IF HU=5 THEN PRINT "You have beate
n me"
  810 IF CO=5 THEN PRINT "This victory i
s the first step in our"'"plan to take o
ver the world!"
```

```
  820 IF Q=500 THEN PRINT "I accept your
 wish to concede"
  830 IF Q1=500 THEN PRINT "I concede th
e game to Master"
  840 END
  850 ENDPROC

  860 DEF PROCinit
  870 X=0
  880 Q1=0
  890 Q=0
  900 CLS
  910 VDU 23;8202;0;0;0
  920 HU=0
  930 CO=0
  940 DIM H(99),Z(8)
  950 FOR A=1 TO 99
  960 IF A>77 OR A=70 OR A=60 OR A=68 OR
 A=69 OR A=50 OR A=59 OR A=58 OR A=40 OR
 A=49 OR A=48 THEN 1010
  970 IF A=30 OR A=38 OR A=39 OR A=20 OR
 A=28 OR A=29 OR A=18 OR A=19 OR A<11 TH
EN 1010
  980 H(A)=255
  990 IF A>70 AND A<78 H(A)=67
 1000 IF A>10 AND A<18 H(A)=72
 1010 NEXT
 1020 FOR A=1 TO 8
 1030 READ B
 1040 Z(A)=B
 1050 NEXT
 1060 DATA -8,-21,-12,-19,19,12,21,8
 1070 ENDPROC
```

```
 1080 DEF PROCraspberry
 1090 PRINT CHR$(11);CHR$(11);CHR$(129);
"No,no,no!! Move the piece on"
 1100 FOR N=1 TO RND(24)
 1110 SOUND 1,-15,100-4*N,1
 1120 NEXT
 1130 ENDPROC

 1140 DEF PROCscore
 1150 PRINT CHR$(130);"  My score is ";C
O;CHR$(131);"   Yours is ";HU
 1160 ENDPROC

 1170 DEF PROCjoy
 1180 FOR J=255 TO 200 STEP -5
 1190 SOUND 3,-15,J,3
 1200 NEXT
 1210 ENDPROC
```

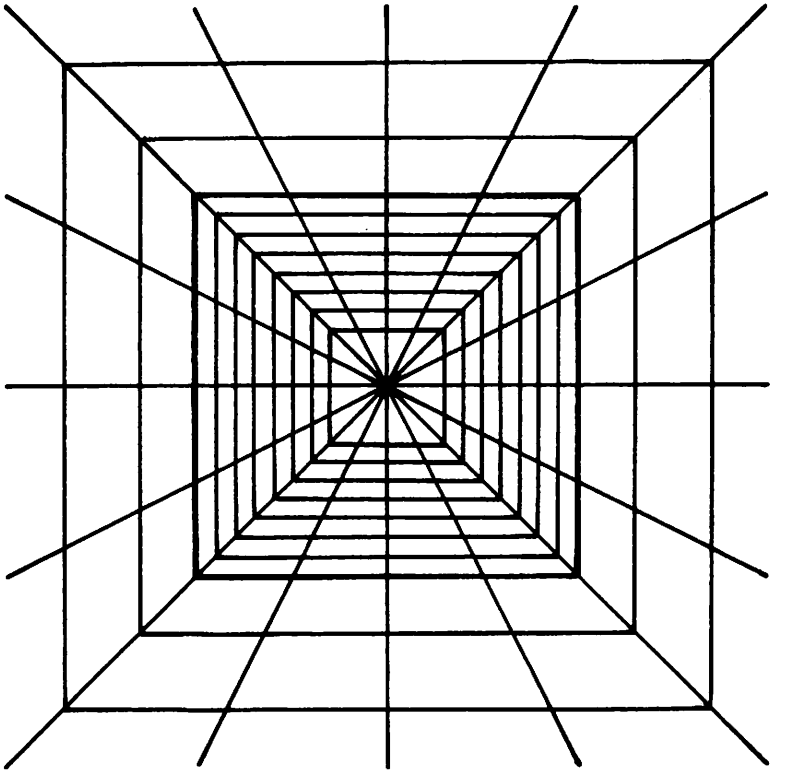


GCOL — TUNNEL

This program shows how the GCOL statement can make a tunnel appear on the screen. First, lines are drawn from the centre to the sides of the screen, using a random GCOL statement; then lines are drawn outwards from the centre of the screen, not quite reaching the sides, using a different colour and a different GCOL statement, so that a different pattern appears. This is done again and again until it looks as if a series of squares is coming out from the centre, while the length of the lines proceeding from the centre keeps on decreasing.

```
 10 MODE 2
 20 VDU 19,0,4,0,0,0
 30 VDU 29,640;512;

 40 REPEAT
 50   FOR B=500 TO 50 STEP -42
 60     F=RND(15)-1
 70     GCOL F*RND(10),F
 80     FOR A=1 TO B*2 STEP 12
 90       PROCA
100       DRAW B-A,-B
110       PROCA
120       DRAW -B,B-A
130       PROCA
140       DRAW B-A,B
150       PROCA
160       DRAW B,B-A
```

```
170         NEXT
180       NEXT
190    UNTIL FALSE

200 DEF PROCA
210 MOVE 0,0
220 ENDPROC
```



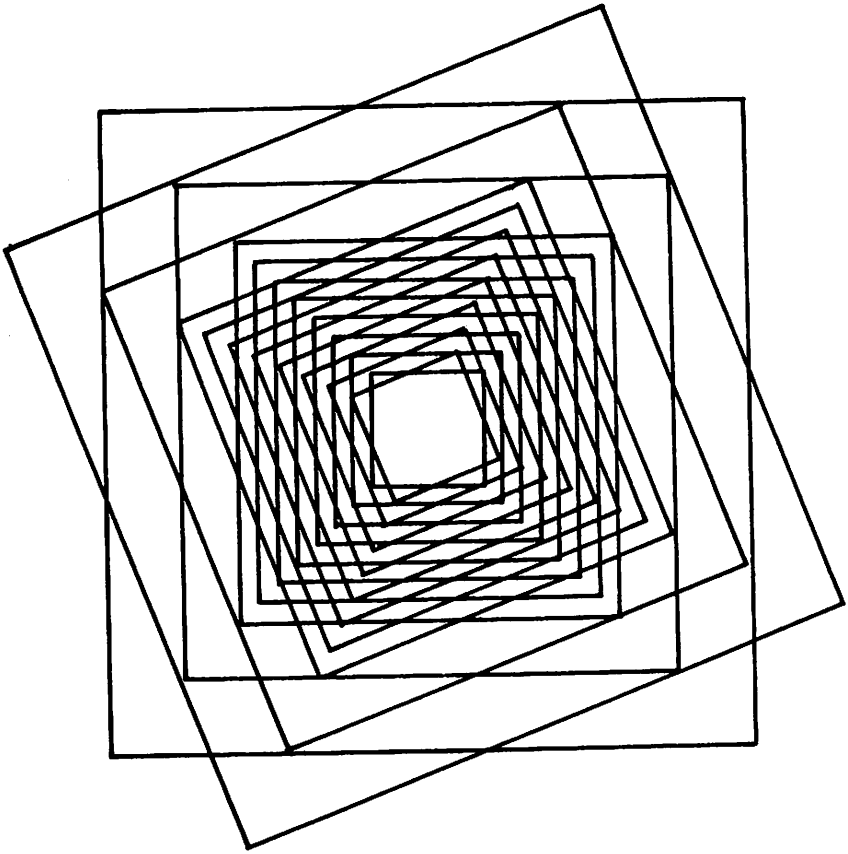
MOIRE PATTERN

'Moire' is defined as a series of patterns occuring when a sheet printed with one family of regular curves or lines is superimposed on another. The patterns change strikingly as the separate sheets are moved slightly.

This effect is produced by drawing a set of lines from the centre to the sides of the screen, then moving that centre slightly and drawing further lines to the side of the screen. This is done five times and then one of the statements in the program — the GCOL statement — is changed.

```
  5 FOR D=0 TO 20
 10   MODE 1
 11   FOR C=1 TO 5
 12     GCOL D,2
 15     FOR B=0 TO 1
 20       FOR A=1 TO 1000 STEP 25
 30         PROCA
 40         DRAW 1140-A,0
 50         PROCA
 60         DRAW 140,A
 70         PROCA
 80         DRAW 140+A,1000
 90         PROCA
100         DRAW 1140,1000-A
110         NEXT
115       GCOL D,1
120       NEXT
```

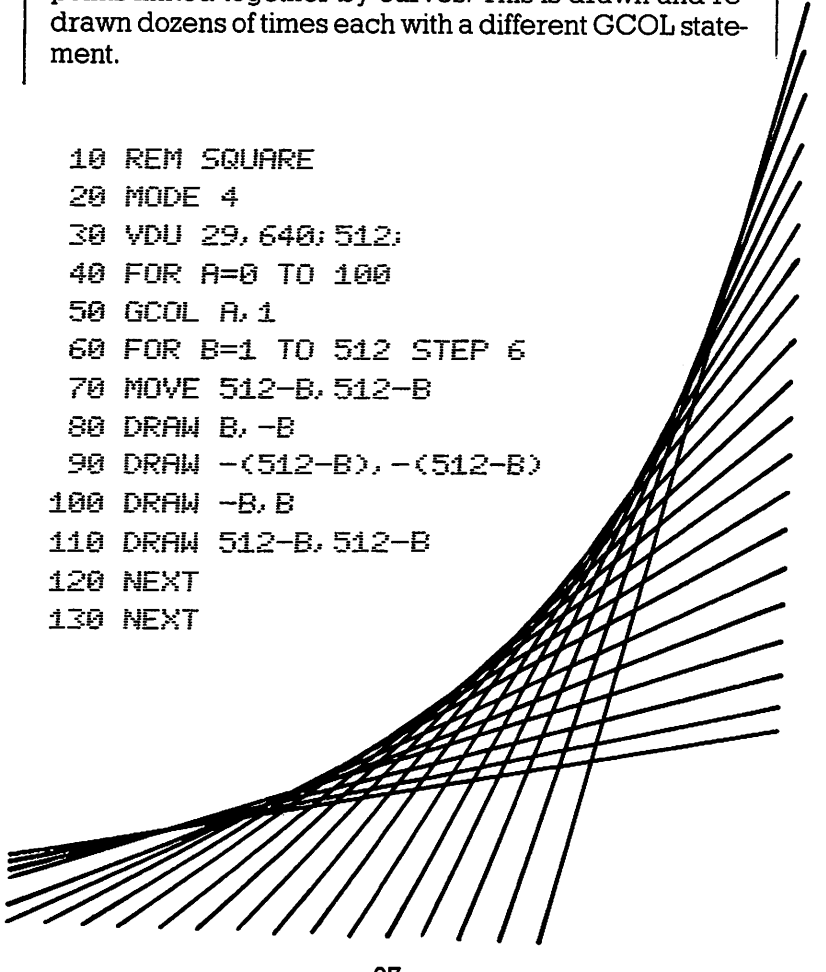
```
130      NEXT
140    NEXT
150 DEF PROCA
160 MOVE 600+B*10,512
170 ENDPROC
```



SQUARE

This program shows how curves can be drawn by using many straight lines. Using this idea, a beautiful pattern of intersecting lines is drawn from the centre of the screen, so that it looks as if you have a four-pointed star with the points linked together by curves. This is drawn and re-drawn dozens of times each with a different GCOL statement.

```
 10 REM SQUARE
 20 MODE 4
 30 VDU 29,640;512;
 40 FOR A=0 TO 100
 50 GCOL A,1
 60 FOR B=1 TO 512 STEP 6
 70 MOVE 512-B,512-B
 80 DRAW B,-B
 90 DRAW -(512-B),-(512-B)
100 DRAW -B,B
110 DRAW 512-B,512-B
120 NEXT
130 NEXT
```



INVADER BLASTER

The aim of this game is to destroy the space invaders who appear above the runway you see in front of you. You are controlling a galactic patrol ship which has left and right, and up and down controls. As you move your controls down, the horizon moves up, and vice-versa.

The runway in front of you shows where the target is going to appear. You know it will be appearing about 600 metres above the runway. The crafty space invaders, however, create bogus UFOs which appear all over the screen — you have to get the real space invader.

On the screen you are given your **x, y,** and **z** co-ordinates. Your **y** co-ordinate gives your height above the ground in metres. Your **x** co-ordinate gives your distance away from the centre of the runway in metres. Your **z** co-ordinate tells you how far you've gone.

If it takes you 10,000 metres (10 km) to destroy the alien, you will be blown up by the invisible mines it lays; because you are travelling at about 100 metres per second, it means you've got just over 1½ minutes to destroy the alien. Your controls are 'Y' to go up, 'B' to go down, 'Z' to go left and 'X' to go right. To fire, you press the space bar.

The problem is that at the high speed you are travelling your velocity from left to right and up and down is continually changing because of wind resistance. You must surmount these difficulties by deft use of your controls.

```
 10 MODE 5
 20 PROCinit
 30 PROCchr
 40 PROCscreen
 50 VDU 19,0,4,0,0,0
 60 TIME=0
 70 REPEAT
 80 P%=RND(10)-5
 90 M%=RND(30)-15
100 VDU 29,0;0;
110 VDU 24,8;Q%;1270;900;
120 GCOL 0,129
130 CLG
140 VDU 24,8;400;1270;Q%;
150 GCOL 0,128
160 CLG
170 VDU 29,640+S%;0;
180 IF P%>2 THEN VDU 31,9,12,244,245
190 Z%=INKEY(0)
200 IF Z%=89 Q%=Q%-4
210 IF Z%=66 Q%=Q%+4
220 IF Z%=90 THEN S%=S%+14
230 IF Z%=88 THEN S%=S%-14
240 IF Z%=32 THEN PROCfire
250 IF Q%<404 THEN Q%=Q%+8
260 IF Q%>899 THEN Q%=Q%-8
270 SOUND 17,-15,S%/3+128,2
280 SOUND 18,-15,S%/3+130,2
290 SOUND 18,-15,Q%/3,2
300 SOUND 19,-15,Q%/3+2,2
310 S%=S%+M%
320 Q%=Q%+(P%/3)
```

```
330 MOVE S%+50,0%
340 DRAW 400,400
350 MOVE -400,400
360 DRAW -(50-S%),0%
370 VDU 31,0,26,120,246,247,248
380 VDU 31,11,26,121,246,247,248
390 VDU 31,5,1,122,246,247,248
400 PRINT TAB(5,26);S%;" "
410 PRINT TAB(16,26);0%
420 PRINT TAB(10,1);TIME
430 P%=76
440 IF TIME>10000 THEN PROCwham
450 UNTIL 0
460 DEF PROCchr
470 DATA 02,02,02,02,02,02,FE,00
480 DATA 40,40,40,40,40,40,7F,00
490 DATA 00,FE,02,02,02,02,02,02
500 DATA 00,7F,40,40,40,40,40,40
510 DATA 00,03,17,2A,1F,0F,01,00
520 DATA 00,80,D0,A8,F0,E0,00,00
530 DATA 00,00,00,77,45,77,00,00
540 DATA 00,00,00,0E,6A,0E,00,00
550 DATA 00,00,00,EC,8A,8E,00,00
560 FOR V%=240 TO 248
570 VDU 23,V%
580 FOR W%=0 TO 7
590 READ A$
600 VDU EVAL("&"+A$)
610 NEXT
620 NEXT
630 ENDPROC
640 DEF PROCfire
```

```
650 A%=TRUE
660 VDU 31,9,12,240,241,31,9,13,242,243
670 VDU 29,640;0;
680 GCOL 0,2
690 MOVE -1600,0
700 DRAW 0,600
710 MOVE 1600,0
720 DRAW 0,600
730 FOR SA=100 TO 200
740 SOUND 16,-15,7,10
750 SOUND 17,00,SA,10
760 NEXT
770 IF Q%<610 OR 0%>590 Z%=0
780 IF Z%=0 AND P%>2 THEN PROCbang
790 VDU 29,640+S%;0;
800 ENDPROC
810 DEF PROCbang
820 IF S%>10 OR S%<-10 ENDPROC
830 VDU 19,1,15,0,0,0
840 VDU 26
850 SOUND 0,-15,7,255
860 GCOL 4,1
870 VDU 29,640;650;
880 FOR W=1 TO 150
890 PLOT 69,RND(200)-100,RND(200)-100
900 NEXT
910 FOR X=1 TO 150
920 PLOT 69,RND(400)-200,RND(400)-200
930 NEXT
940 FOR Y=1 TO 150
950 PLOT 69,RND(800)-400,RND(800)-400
960 NEXT
```

```
 970 FOR Z=1 TO 550
 980 PLOT 69,RND(1600)-800,RND(1600)-800
 990 NEXT
1000 IF A% REPEAT UNTIL 0
1010 ENDPROC
1020 DEF PROCscreen
1030 MOVE 1,910
1040 DRAW 1279,910
1050 DRAW 1279,390
1060 DRAW 1,390
1070 DRAW 1,910
1080 FOR A=000 TO 1300 STEP 100
1090 MOVE A,300
1100 DRAW A,390
1110 NEXT
1120 FOR A=025 TO 1275 STEP 50
1130 MOVE A,300
1140 DRAW A,325
1150 NEXT
1160 FOR A=050 TO 1250 STEP 50
1170 MOVE A,300
1180 DRAW A,350
1190 NEXT
1200 MOVE 312,200
1210 DRAW 628,200
1220 DRAW 640,188
1230 DRAW 640,162
1240 DRAW 628,150
1250 DRAW 312,150
1260 DRAW 300,162
1270 DRAW 300,188
1280 DRAW 312,200
```

```
1290 MOVE 1012,200
1300 DRAW 1228,200
1310 DRAW 1240,188
1320 DRAW 1240,162
1330 DRAW 1228,150
1340 DRAW 1012,150
1350 DRAW 1000,162
1360 DRAW 1000,188
1370 DRAW 1012,200
1380 MOVE 25,240
1390 DRAW 1255,240
1400 DRAW 1255,370
1410 DRAW 25,370
1420 DRAW 25,240
1430 MOVE 0,220
1440 DRAW 1280,220
1450 MOVE 0,130
1460 DRAW 1280,130
1470 MOVE 0,300
1480 DRAW 1280,300
1490 MOVE 640,0
1500 DRAW 640,130
1510 MOVE 320,0
1520 DRAW 320,130
1530 MOVE 980,0
1540 DRAW 980,130
1550 MOVE 0,0
1560 DRAW 1279,0
1570 DRAW 1279,1023
1580 DRAW 0,1023
1590 DRAW 0,0
1600 GCOL 0,3
```

```
1610 MOVE 640,914
1620 DRAW 640,950
1630 MOVE 0,950
1640 DRAW 1280,950
1650 ENDPROC
1660 DEF PROCinit
1670 M%=0
1680 Z%=0
1690 M=TRUE
1700 R%=1
1710 Q%=500
1720 S%=0
1730 *FX  9,5
1740 *FX 10,5
1750 ENDPROC
1760 DEF PROCwham
1770 S%=0
1780 A%=FALSE
1790 PROCbang
1800 REPEAT
1810 FOR X=0 TO 19
1820 VDU 23,0,13,X,0,0,0,0,0,0
1830 NEXT
1840 FOR X=19 TO 0 STEP -1
1850 VDU 23,0,13,X,0,0,0,0,0,0
1860 NEXT
1870 UNTIL 0
1880 ENDPROC
```

PAC-TRAC

In this game your maze-man is off on another adventure. His aim is to eat up all the blocks but he is in some difficulty because, as he moves around, he leaves a trail of blocks he cannot digest and he cannot therefore retrace his path. You have 90 seconds to eat up all the blocks.

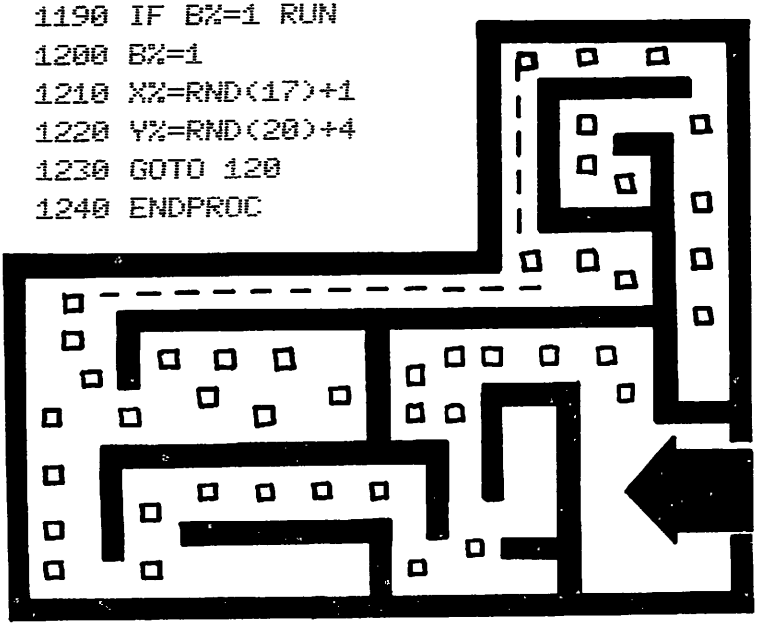
```
  10
  20 REM PAC-TRACK
  30
  40 ON ERROR PROCtele
  50 MODE 5

  60 PROCchr
  70 PROCinit

  80 FOR A=1 TO 100
  90 VDU 31,RND(17)+1,RND(20)+4,232.
 100 NEXT

 110 TIME=0
 120 REPEAT
 130 PROCmove
 140 UNTIL TIME>9000

 150 *FX 4
 160 *FX 12

 170 PRINT TAB(0,23)"YOU HAVE EATEN ";P
%;" PILLS"
```

```
  180 PRINT TAB(0,25)"  ANY KEY TO START
"
  190 S=GET
  200 RUN

  210 DEF PROCmove
  220 LOCAL A%
  230 VDU 17,2,31,X%,Y%,Z%
  240 A%=INKEY(5%)
  250 IF A%<136 OR A%>139 THEN A%=0%
  260 *FX 15,1
  270 *FX 15,6
  280 PRINT TAB(0,1)"PILLS:";P%;TAB(10);
"TIME:"TIME/100
  290 IF A%=139 THEN PROCup
  300 IF A%=138 THEN PROCdown
  310 IF A%=137 THEN PROCright
  320 IF A%=136 THEN PROCleft
  330 Q%=A%
  340 VDU 31,X%,Y%,Z%
  350 ENDPROC

  360 DEF PROCup
  370 C%=FNread(X%,Y%-1)
  380 IF C%=225 THEN A%=Q% : ENDPROC
  390 VDU 31,X%,Y%,225
  400 Y%=Y%-1
  410 IF C%=232 THEN P%=P%+1 : SOUND 1,-
15,10,2 ELSE SOUND 1,-15,200,2
  420 Z%=229
  430 ENDPROC
  440 DEF PROCdown
```

```
  450 C%=FNread(X%,Y%+1)
  460 IF C%=225 THEN A%=0% : ENDPROC
  470 VDU 31,X%,Y%,225
  480 Y%=Y%+1
  490 IF C%=232 THEN P%=P%+1 : SOUND 1,-
15,10,2 ELSE SOUND 1,-15,200,2
  500 Z%=230
  510 ENDPROC
  520 DEF PROCright
  530 C%=FNread(X%+1,Y%)
  540 IF C%=225 THEN A%=0% : ENDPROC
  550 VDU 31,X%,Y%,225
  560 X%=X%+1
  570 IF C%=232 THEN P%=P%+1 : SOUND 1,-
15,10,2 ELSE SOUND 1,-15,200,2
  580 Z%=227
  590 ENDPROC
  600 DEF PROCleft
  610 C%=FNread(X%-1,Y%)
  620 IF C%=225 THEN A%=0% : ENDPROC
  630 VDU 31,X%,Y%,225
  640 X%=X%-1
  650 IF C%=232 THEN P%=P%+1 : SOUND 1,-
15,10,2 ELSE SOUND 1,-15,200,2
  660 Z%=228
  670 ENDPROC
  680 DEF FNread(X%,Y%)
  690 LOCAL A%,C%
  700 A%=135
  710 VDU 31,X%,Y%
  720 C%=((USR(&FFF4) AND &FFFF) DIV &10
0)
```

```
730 IF C%<128 THEN=C%
740=(C% MOD 32)+224
750 DEF PROCchr
760 LOCAL T%,G%
770 DATA 00,7E,7E,7E,7E,7E,7E,00
780 DATA 3C,7E,FF,FF,FF,FF,7E,3C
790 DATA 3C,7C,F8,F0,F0,F8,7C,3C
800 DATA 3C,3E,1F,0F,0F,1F,3E,3C
810 DATA 00,00,C3,E7,FF,FF,7E,3C
820 DATA 3C,7E,FF,FF,E7,C3,00,00
830 DATA 18,3C,7E,5A,7E,66,7E,5A
840 DATA 00,00,00,18,18,00,00,00
850 FOR T%=225 TO 232
860 VDU 23,T%
870 FOR G%=0 TO 7
880 READ A$
890 VDU EVAL("&"+A$)
900 NEXT
910 NEXT
920 ENDPROC
930 DEF PROCinit
940 VDU 23;8202;0;0;0;
950 VDU 19,0,4,0,0,0
960 *FX 4,1
970 *FX 12
980 *KEY 0 "*FX 12|M*FX 4|M"
990 @%=&00000090A
1000 X%=RND(5)+8
1010 Y%=RND(10)+8
1020 Z%=227
1030 P%=0
1040 B%=0
```

```
 1050 VDU 30
 1060 PRINT''''STRING$(20,CHR$(225))
 1070 FOR S=4 TO 25
 1080 VDU 31,0,S,225,32,32,32,32,32,32,3
2,32,32,32,32,32,32,32,32,32,32,32,225
 1090 NEXT
 1100 PRINT STRING$(20,CHR$(225))
 1110 VDU 30
 1120 INPUT"SKILL LEVEL(1-5)",S%
 1130 S%=50-S%*10
 1140 PRINT TAB(0,0)"                    "
 1150 *FX 12,1
 1160 *FX 11,1
 1170 ENDPROC

 1180 DEF PROCtele
 1190 IF B%=1 RUN
 1200 B%=1
 1210 X%=RND(17)+1
 1220 Y%=RND(20)+4
 1230 GOTO 120
 1240 ENDPROC
```



How To Write Better Programs

By Tim Hartnell, series editor

There are a number of fine programs in this book, and many of the regular computer magazines contain other such ones. But no matter how good the programs from published sources are, you are certain to get more pleasure from running them if they have been partially or completely written by you. Putting your personal stamp on programs, altering them to reflect your wishes and creativity, is an excellent way to improve the programs, and eventually, of course, you'll become a better and more imaginative programmer.

Programs in magazines, and in books like this one, are ideal as starting points for your own developments. You may also find that advertisements for software packages can be fruitful 'idea-starters'. You only need to read the description of what the commercially available program does, and you will have the first step towards creating your own program. You have to be careful, of course, not to infringe copyright either in the screen displays, in the name of the program, or the names of the 'characters' within the program. However, you will probably find that at a certain point in its development the program will take on a life of its own, growing and evolving away from the original scenario, until you eventually have a completely new game concept and implementation.

Whatever you do, be careful not to pass off other people's work as your own. By all means adapt and improve published programs, but do not then present them to magazines as if they were originals. I have lost count of

the number of times one of my own programs, from one of my books, has been submitted to me for publication.

Always watch out for new ideas as you look through books, game and computer magazines, or wander through video game arcades. It may be worth keeping notes of ideas you come across for games, for character shapes, for sounds, for dramatic endings and so on. Thus you will never be short of ideas, and you will also be able to merge the material together to produce better games which hold the player's attention for longer.

Games tend to fall into one of three categories, and it is worth making sure of the category into which your proposed program will fall *before* you start to program, since the category of game materially alters the programming approach. This is not to say that, as you develop a program, it will not move from one category into another, nor that a particular game might not extend across two categories, but it is nevertheless useful to keep the various groups separate in your mind, just to clarify your thoughts. The three categories are:

1. Board games
2. 'Arcade' (that is, highly visual, fast moving, noisy, real time) games
3. Games of chance (such as Roulette and Snap).

In board games, the quality of play is more important than lightning-fast response, while the arcade-type programs must be kept moving at all costs, even if some 'intelligence' from your Martian intruders must be sacrificed to achieve this. Games of chance depend more on their ease of play ('user-friendly' inputs), and an approach to true randomness, than do either of the other categories.

You will find that games programs tend to fall into types, which are subdivisions of the three above mentioned categories. Many board games are variants of chess or checkers; many arcade games started off life as Space Invader-type games; and games of chance started off in the 'real world' of dice and cards. Looking at a program description, or a games machine, and trying

to categorise the game you see can help trigger new ideas which fit within that particular game's genre.

There is a school of thought within programming — generally called 'structured programming' — which believes that discipline at the beginning of the games-writing process is essential. While less interesting than sitting down at the computer right away, a much better program is produced in the end. I once wrote a program called Dome Dweller, a simulation program in which the player is in charge of a 'lunar dome' and must decide which products to manufacture and sell in order to buy oxygen and food for the station's inhabitants. (This program was used in my book *The Book of Listings*, written with Jeremy Ruston, and published by the BBC.)Once I had decided the overall scenario, I worked out the screen display, and came up with an idea as follows:

> Oxygen supplies are low
> There are 96 people living within your dome in year 3
> Money credit is $5,693
> Annual maintenance charge is $226
> Oxygen tanks hold 811 units
> Oxygen costs $8 per unit
> Each dome dweller needs 5 units a year
> Food stocks stand at 2122
> Each dweller needs 3 units a year ($6 each, $576 for dome. This will last 7 years at present population.)
> You can trade your unique lunar sculptures with the people who live in other domes. You use up 2 units of oxygen making each one, and sell them for $30.

As you can probably guess from this 'sample printout', the idea of the program is to decide how many 'unique lunar sculptures' you must make and sell in order to buy oxygen and food, and to pay the 'annual maintenance' charge. The problem with this particular program is that making each sculpture uses up oxygen, so you must balance your wish to make money against the need to use the oxygen intelligently.

You may well wish to try writing such a program yourself. You should end up with an enjoyable program, and writing it will do much to help you develop your programming skills. The first thing to do is to make a list of what the program has to do:

Set up the needed variables
Tell the player the 'state of the dome'
Ask how much oxygen to be bought
Check if can afford this, if so buy it, if not go back and ask again
Ask how much food to be bought
Check if can afford this, if so buy it, if not go back and ask again
Update oxygen quantity
Update food quantity
Reduce money left total
Ask how many items of sculpture to be made
Check if there is enough oxygen to make this many, if not go back and ask again
Reduce oxygen quantity by amount needed to make the number of sculptures specified, increase money total to reflect value of sculptures made
Increase the population total slightly, add one to the 'current year'
Check if there is enough food in stocks to feed whole population
Check if there is enough oxygen for whole population
Check if there is any money
If any of these conditions are negative (eg not enough food) send action to an 'end of game' routine
If all are positive, loop back to tell the player the state of the dome, and continue to circle

You could probably write a Dome Dweller program using the list above, together with the 'sample printout' information. There is, however, a secret I should like to share with you which unlocks programming problems

almost instantly. You can actually write all the vital parts of a program in minutes, so you can see the raw framework of a program like this running long before you fill in the details. And once you have a framework you can work on it for as long as you like, knowing as you do so that — at every moment in program development — you have a working program. You do not have to wait until the end until you can run it to see how you are going. The 'secret' is to hold the entire program within a series of subroutine calls, all held within a perpetual loop. Here's how it could work with this program. The very first lines you enter in your computer are as follows:

```
10 REM DOME DWELLER
20 GOSUB 1000: REM ASSIGN VARIABLES
30 GOSUB 2000: REM PRINT OUT STATE OF
DOME
40 GOSUB 3000: REM OXYGEN
50 GOSUB 4000: REM FOOD
60 GOSUB 5000: REM SCULPTURE
70 GOSUB 6000: REM UPDATE POPULATION
80 GOSUB 7000: REM CHECK ON STATE OF
 DOME
90 IF (all conditions positive, from GOSUB 7000)
THEN GOTO 30
100 REM End of game ...
```

As you can see once you have the 'master loop' set up in this way, it is relatively simple to fill in each of the subroutines one by one, testing each as you do so, and elaborating each one so that you end up eventually with a very good program. The only thing you need now is a list of the variables which you will use with the program.

I find the best way to do this is to use explicit names for variables so that when you are programming you do not have to spend time checking, for example, whether AA stands for the population, or the number of units of oxygen used up in making each item of sculpture. To make programs as easy as possible to transfer between different computers you can stick to two letter variable names, or you can take advantage (if your computer

allows it) of long names (such as OXYUSE for the amount of oxygen used) for variables. Then you have no doubts whatsoever as to the meaning of each variable name. To show how this can work, and to illustrate a further advantage of explicit variable names, here are the variables used in Dome Dweller:

FOLK — population of dome
CASH — money in treasury
FOOD — food stocks on hand
FOODCOST — how much each unit of food costs
FOODNEED — how many units of food were consumed per person per year
ARTCOST — how much oxygen was used up making each piece of sculpture
ARTPAY — how many dollars each piece of sculpture was sold for
OXY — oxygen stocks on hand
OXYNEED — how many units of oxygen were consumed per person per year
OXYCOST — how much each unit of oxygen cost to buy
REPAIR — the cost of annual repairs to the dome
YEAR — the year of the dome's life

Using explicit variable names in this way — although they use up more memory than do single or double-letter variable names — makes it very simple to follow through a program, working out what each section of the program actually does. Moreover, and this is the further advantage mentioned, it is very easy when writing the program to insert the formulae required for calculations. By this I mean that if, for example, you wished to include (as I do in this program) an indication of how much oxygen is needed for each year, you simply multiply the number of people in the dome (FOLK) by the number of oxygen units each person needs each year (OXYNEED). You can then include within the printouts for the state of the dome a line like:

```
PRINT "THERE ARE ";FOLK;" IN THE DOME"
PRINT "IN YEAR ";YEAR
```

```
PRINT "EACH PERSON NEEDS ";OXYNEED;"
      UNITS OF"
PRINT "OXYGEN EACH YEAR,";
      OXYNEED*FOLK;" NEEDED"
PRINT "FOR THE WHOLE DOME"
```

It also makes it very easy to check on whether purchases are possible. For example, to buy food, you could say:

```
PRINT "HOW MUCH FOOD WILL YOU BUY?"
INPUT A
IF A*FOODCOST > CASH THEN GOTO (get
another A)
```

Thus to improve your programs by the use of 'structured programming' you should: draw up a sample printout, or mock-up of the final screen display; draw up a list of what the program has to do each time through a 'master control loop'; change this list to a series of subroutine calls; use explicit variable names if possible.

When designing programs for others ensure that it is clear what the player should do when running the program. There is little point, when memory is limited, in including a long set of instructions within the program, but you should write such instructions down. User prompts should be explicit (such as ENTER THE NUMBER OF GOES YOU WANT) and should include warnings of the limits which will be placed on the input (HOW MANY CARDS WILL YOU START WITH: 1, 2 OR 3 ?, for instance).

You cannot assume that you will be present every time a program is run, so you should do your best to make it as foolproof as possible. If you can, add error-trapping routines to the program to ensure that a mistake in entering a choice earlier on in the program will not cause it to crash or come up with stupid results later on.

If you read through this section of the book several times and try to apply the ideas to your own programming work, you should find your work quality improves significantly, and also that you can spend more time improving and embellishing a program and less in the raw mechanical task of getting the thing running.

GLOSSARY

A

Accumulator — the place within the computer in which arithmetic computations are performed and where the results of these computations are stored.

Algorithm — the series of steps the computer follows to solve a particular problem.

Alphanumeric — this term is usually used in relation to a keyboard, as in 'it is an alphanumeric keyboard', which means that the keyboard has letters as well as numbers. It is also used to refer to the 'character set' of the computer. The character set comprises the numbers and letters the computer can print on the screen.

ALU (Arithmetic/Logic Unit) — the part of the computer which does arithmetic (such as addition, subtraction) and where decisions are made.

AND — a Boolean logic operation that the computer uses in its decision-making process. It is based on Boolean algebra, a system developed by mathematician George Boole (1815-64). In Boolean algebra the variables of an expression represent a logical operation such as OR and NOR.

ASCII — stands for American Standard Code for Information Exchange, the most widely used encoding system for English language alphanumerics. There are 128 upper and lower case letters, digits and some special characters. ASCII converts the symbols and control instructions into seven-bit binary combinations.

Assembler — a program which converts other programs written in assembly language into machine code (which the computer can understand directly). Assembly language is a low level programming language which uses easily memorised combinations of two or three letters to represent a particular instruction which the assembler then converts so the machine can understand it. Examples of these are ADD (add), and SUB (subtract). A computer programmed in assembly language tends to work more quickly than one programmed in a higher level language such as BASIC.

B

BASIC — an acronym for Beginners All-Purpose Symbolic Instruction Code. It is the most widely used computer language in the microcomputer field. Although it has been criticised by many people, it has the virtue of being very easy to learn. A great number of BASIC statements resemble ordinary English.

Baud — named after Baudot, a pioneer of telegraphic communications. Baud measures the rate of transfer of information and is approximately equal to one bit per second.

BCD — an abbreviation for Binary Coded Decimal.

Benchmark — a test against which certain functions of the computer can be measured. There are a number of so-called 'standard Benchmark tests', but generally these only test speed. This is rarely the aspect of a microcomputer that is most of interest to the potential buyer.

Binary — a numbering system that uses only zeros and ones.

Bit — an abbreviation for Binary Digit. This is the smallest unit of information a computer circuit can recognise.

Boolean Algebra — the system of algebra developed by mathematician George Boole which uses algebraic notation to express logical relationships (see AND).

Bootstrap — a short program or routine which is read into the computer when it is first turned on. It orients the computer to accept the longer, following program.

Bug — an error in a computer program which stops the program from running properly. Although it is generally used to mean only a fault or an error in a program, the term bug can also be used for a fault in the computer hardware.

Bus — a number of conductors used for transmitting signals such as data instructions, or power in and out of a computer.

Byte — a group of binary digits which make up a computer word. Eight is the most usual number of bits in a byte.

C

CAI — Computer Assisted Instruction.

CAL — Computer Assisted Learning. The term is

generally used to describe programs which involve the learner with the learning process.

Chip — the general term for the entire circuit which is etched onto a small piece of silicon. The chip is, of course, at the heart of the microcomputer.

Clock — the timing device within the computer that synchronises its operations.

COBOL — a high level language derived from the words Common Business Orientated Language. COBOL is designed primarily for filing and record-keeping.

Comparator — a device which compares two things and produces a signal related to the difference between the two.

Compiler — a computer program that converts high level programming language into binary machine code so the computer can handle it.

Complement — a number which is derived from another according to specified rules.

Computer — a device with three main abilities or functions:
1) to accept data
2) to solve problems
3) to supply results

CPU — stands for Central Processing Unit. This is the heart of the computer's intelligence, where data is handled and instructions are carried out.

Cursor — a character which appears on the TV screen when the computer is operating. It shows where the next character will be printed. On a computer there are usually 'cursor control keys' to allow the user to move the cursor around the screen.

D

Data — information in a form which the computer can process.

Debug — the general term for going through a program and correcting any errors in it, that is, chasing down and removing bugs (see Bug).

Digital Computer —a computer which operates on information which is in a discrete form.

Disk/Disc — this is a magnetically sensitised plastic disk, a little smaller than a single play record. This is used for

storing programs and for obtaining data. Disks are considerably faster to load than a cassette of the same length program. The disk can be searched very quickly while a program is running for additional data.

Display — the visual output of the computer, generally on a TV or monitor screen.

Dot Matrix Printer — a printer which prints either the listing of a program or that which is displayed on the TV screen. Each letter and character is made up of a number of dots. The higher the number of dots per character the finer the resolution of the printer.

Dynamic Memory — a memory unit within the computer which 'forgets' its contents when the power is turned off.

E

Editor — this term is generally used for the routine within the computer which allows you to change lines of a program while you are writing it.

EPROM — stands for Erasable Programmable Read-Only Memory. This is like the ROM in the computer, except that it is fairly easy to load material into an EPROM and it doesn't disappear when you turn the power off. EPROMs must be placed in a strong ultra violet light to erase them.

Error Messages — the information given by a computer where there is a fault in the coding during a part of a program, usually shown by the computer stopping, and printing a word, or a word and numbers, or a combination of numbers only, at the bottom of the screen. This tells you what mistake has been made. Common mistakes include using the letter O instead of zero in a line, or leaving out a pair of brackets, or one of the brackets, in an expression, or failing to define a variable.

F

File — a collection of related items of information organised in a systematic way.

Floppy Disk — a relatively cheap form of magnetic disk used for storing computer information, and so named because it is quite flexible (see Disk/Disc).

Flow Chart — a diagram drawn up before writing a program, in which the main operations are enclosed within

rectangles or other shapes and connected by lines, with arrows to represent loops, and decisions written at the branches. It makes writing a program much easier because traps such as infinite loops, or non-defined variables can be caught at an early stage. It may not be worth writing a flow chart for very short programs, but generally a flow chart aids in creating programs.

Firmware — there are three kinds of 'ware' in computers: software 'temporary' programs; hardware like the ROM which contains permanent information; and firmware in which the information is relatively permanent, as in an EPROM (see EPROM).

Flip-Flop — a circuit which maintains one electrical condition until changed to the opposite condition by an input signal.

FORTRAN — an acronym for FORmula TRANslation, this is a high level, problem orientated computer language for scientific and mathematical use.

G

Gate — an electrical circuit which, although it may accept one or more incoming signals, only sends out a single signal.

Graphics — pictorial information as opposed to letters and numbers.

H

Hard Copy — computer output which is in permanent form.

Hardware — the physical parts of the computer (also see software and firmware).

Hexadecimal (Hex) — a numbering system to the base sixteen. The digits zero to nine are used, as well as the letters A, B, C, D, E and F to represent numbers. A equals 10, B equals 11, C equals 12, and so on. Hex is often used by microprocessor users.

Hex Pad — a keyboard designed specifically for entering hexadecimal notation.

High Level Language — a programming language which allows the user to talk to the computer more or less in English. In general, the higher the level of the language (that is, the

closer it is to English), the longer it takes for the computer to translate it into a language it can use. Lower level languages are far more difficult for human operators but are generally executed far more quickly.

I

Input — the information fed into the computer via a keyboard, a microphone, a cassette or a disk.

Input/Output (I/O Device) — a device which accepts information or instructions from the outside world, relays it to the computer, and then, after processing, sends the information out in a form suitable for storing, or in a form which could be understood by a human being.

Instruction — data which directs a single step in the processing of information by the computer (also known as a command).

Integrated Circuit — a complete electronic circuit imprinted on a semiconductor surface.

Interface — the boundary between the computer and a peripheral such as a printer.

Interpreter — a program which translates the high level language fed in by the human operator, into a language which the machine can understand.

Inverter — a logic gate that changes the signal being fed in, to the opposite one.

Interactive Routine — part of a program which is repeated over and over again until a specified condition is reached.

J

Jump Instruction — an instruction which tells the computer to go to another part of the program, when the destination of this move depends on the result of a calculation just performed.

K

K — this relates to the size of the memory. Memory is usually measured in 4K blocks. 1K contains 1,024 bytes.

Keyword — the trigger word in a line of programming, usually the first word after the line number. Keywords include STOP, PRINT and GOTO.

L

Language — computer languages are divided into three sections: high level languages, such as BASIC, which are reasonably close to English and fairly easy for humans to use; low level languages, such as Assembler, that use short phrases which have some connection with English (ADD for add and RET for return, for instance); and machine code which communicates more or less directly with the machine.

LCD — this stands for Liquid Crystal Diode. Some computers such as the TRS-80 Pocket Computer use an LCD display.

LED — this stands for Light Emitting Diode. The bright red numbers which are often used on watch or clock displays are made up of LEDs.

Logic — the mathematical form of a study of relationships between events.

Loop — a sequence of instructions within a program which is performed over and over again until a particular condition is satisfied.

M

Machine Language or Machine Code — an operation code which can be understood and acted upon directly by the computer.

Magnetic Disk — see Disk and Floppy Disk.

Mainframe — computers are generally divided into three groups, and the group a computer falls into depends more or less on its size. The computer you are thinking of buying is a microcomputer; medium sized computers are known as minicomputers; and the giant computers that you sometimes see in science fiction movies are mainframe computers. Until 15 years ago mainframe computers were, in practical terms, the only ones available.

Memory — there are two types of memory within a computer. The first is called ROM (read-only memory); this is the memory that comes already programmed on the

computer, which tells the computer how to make decisions and how to carry out arithmetic operations. This memory is unaffected when you turn the computer off. The second type is RAM (random access memory). This memory holds the program you type in at the keyboard or send in via a cassette or disk. In most computers the computer 'forgets' what is in RAM when you turn the power off.

Microprocessor — the heart of any computer. It requires peripheral unit interfaces, such as a power supply and input and output devices, to act as a microcomputer.

MODEM — stands for Modulator Demodulator. This is a device which allows two computers to talk to each other over the telephone. The computers usually use a cradle in which a telephone receiver is placed.

Monitor — this has two meanings in computer terms. One meaning is a television-like display. A monitor has no facility for tuning television programs, and usually the picture produced on a monitor is superior to that produced by an ordinary television. The second meaning of a monitor relates to ROM. The monitor of a computer is described as the information it has built in when you buy it. This information allows it to make decisions and carry out arithmetic computations.

Motherboard — a framework to which extra circuits can be added. These extra circuits often give the computer facilities which are not built-in, such as that of producing sound or of controlling a light pen.

MPU — an abbreviation for Microprocessor Unit.

N

Nano-second — a nano-second is one thousand billionth of a second, the unit of speed in which a computer or a memory chip is often rated.

Non-Volatile Memory — memory which is not lost when the computer is turned off. Some of the smaller computers such as the TRS-80 Pocket Computer have non-volatile memory. The batteries hold the program you enter for several hundred hours.

Not — a Boolean logic operation that changes a binary digit into its opposite.

Null String — a string which contains no characters. It is shown in the program as two double quote marks, without anything between them.

Numeric — pertaining to numbers as opposed to letters (that is, alphabetic). Many keyboards are described as being alphanumeric which means both numbers and letters are provided.

O

Octal — a numbering system which uses eight as the base, and the digits 0, 1, 2, 3, 4, 5, 6 and 7. The Octal system is not used very much nowadays in microcomputer fields. The Hexadecimal system is more common (see Hexadecimal).

Operating System — the software or firmware generally provided with the machine that allows you to run other programs.

OR — an arithmetic operation that returns a 1, if one or more inputs are 1.

Oracle — a method of sending text messages with a broadcast television signal. A teletext set is required to decode the messages. Oracle is run by Independent Television Service in the UK, and a similar service — Ceefax — is provided by the BBC.

Output — information or data fed out by the computer to such devices as a TV-like screen, a printer or a cassette tape. The output usually consists of the information which the computer has produced as a result of running a program.

Overflow — a number too large or too small for the computer to handle.

P

Pad — see Keypad.

Page — often used to refer to the amount of information needed to fill one TV screen, so you can talk about seeing a page of a program, the amount of the listing that will appear on the screen at one time.

PASCAL — a high level language.

Peripheral — anything which is hooked onto a computer, for control by the computer, such as a disk unit, a printer or a voice synthesiser.

Port — a socket through which information can be fed out of or in to a computer.

Prestel — the British telecom name for a system of calling up pages of information from a central computer via the

telephone and displaying them on a television screen. A similar commercial version in the United States is known as The Source.

Program — in computer terms program has two meanings. One is the list of instructions that you feed into a computer, and the second is used as a verb, as in 'to program a computer'.

PROM — stands for Programmable Read Only Memory. This is a device which can be programmed, and once it is then the program is permanent (also see EPROM and ROM).

R

Random Access Memory (RAM) — the memory within a computer which can be changed at will by the person using the computer. The contents of RAM are usually lost when a computer is turned off. RAM is the memory device that stores the program that you type in and also stores the results of calculations in progress.

Read-Only Memory (ROM) — in contrast to RAM, information in ROM cannot be changed by the user of the computer, and the information is not lost when the computer is turned off. The data in ROM is put there by the manufacturers and tells the computer how to make decisions and how to carry out arithmetic computations. The size of ROM and RAM is given in the unit K (see K).

Recursion — the continuous repetition of a part of the program.

Register — a specific place in the memory where one or more computer words are stored during operations.

Reserved Word — a word that you cannot use for a variable in a program because the computer will read it as something else. An example is the word TO. Because TO has a specific computer meaning, most computers will reject it as a name for a variable. The same goes for words like FOR, GOTO and STOP.

Routine — this word can be used as a synonym for program, or can refer to a specific section within a program (also see Subroutine).

S

Second Generation — this has two meanings. The first applies to computers using transistors, as opposed to first

generation computers which used valves. Second generation can also mean the second copy of a particular program; subsequent generations are degraded by more and more noise.

Semiconductor — a material that is usually an electrical insulator but under specific conditions can become a conductor.

Serial — information which is stored or sent in a sequence, one bit at a time.

Signal — an electrical pulse which is a conveyor of data.

Silicon Valley — the popular name given to an area in California where many semiconductor manufacturers are located.

SNOBOL — a high level language.

Software — the program which is entered into the computer by a user which tells the computer what to do.

Software Compatible — this refers to two different computers which can accept programs written for the other.

Static Memory — a non-volatile memory device which retains information so long as the power is turned on, but does not require additional boosts of power to keep the memory in place.

Subroutine — part of a program which is often accessed many times during the execution of the main program. A subroutine ends with an instruction to go back to the line after the one which sent it to the subroutine.

T

Teletext — information transmitted in the top section of a broadcast television picture. It requires a special set to decode it to fill the screen with text information. The BBC service is known as Ceefax, the ITV service as Oracle. Teletext messages can also be transmitted by cable, for example the Prestel service in Britain or The Source in the United States.

Teletype — a device like a typewriter which can send information and also receive and print it.

Terminal — a unit independent of the central processing unit. It generally consists of a keyboard and a cathode ray display.

Time Sharing — a process by which a number of users may have access to a large computer which switches rapidly

from one user to another in sequence, so each user is under the impression that he or she is the sole user of the computer at that time.

Truth Table — a mathematical table which lists all the possible results of a Boolean logic operation, showing the results you get from various combinations of inputs.

U

UHF — Ultra High Frequency (300-3000 megaHertz).

Ultra Violet Erasing — Ultra violet light must be used to erase EPROMs (see EPROM).

V

Variable — a letter or combination of letters and symbols which the computer can assign to a value or a word during the run of a program.

VDU — an abbreviation for Visual Display Unit.

Volatile — refers to memory which 'forgets' its contents when the power is turned off.

W

Word — a group of characters, or a series of binary digits, which represent a unit of information and occupy a single storage location. The computer processes a word as a single instruction.

Word-Processor — a highly intelligent typewriter which allows the typist to manipulate text, to move it around, to justify margins and to shift whole paragraphs if necessary on a screen before outputting the information onto a printer. Word-processors usually have memories, so that standard letters and the text of letters, written earlier, can be stored.

The Virgin Computer Games Series

GAMES FOR YOUR BBC MICRO

More than 20 challenging programs, each one especially written for the series and guaranteed to provide hours of entertainment.

The games include 3D TERROR CHASE (escape from the spooky ghosts in a haunted castle); REACT (this vivid program tests your split-second reactions); MAZE-MAN (dodge the snipes in the maze before they devour you); PONTOON (watch the computer deal the cards — it never cheats!); MAGIC SQUARE (can you discover the missing numbers?); LIFE-CHANGE UNITS (your very own computerised shrink — 'Is a nervous breakdown round the corner?', now you'll know); CHECKERS (can you beat the computer?).

GAMES FOR YOUR BBC MICRO will improve your programming skills as you follow the instructions to put each of the programs into your machine, and comes complete with a brief dictionary of computer terms and some hints on how to improve and extend the programs in the book.

Programs of originality and quality for all the family.

Virgin

0 907080 83 9

£2.95